MATHEMATICAL MODELING OF BIOLOGICAL CLOCKS

A Dissertation

by

JONATHAN P. TYLER

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Jay Walton
Co-Chair of Committee,	Anne Shiu
Committee Members,	Harold Boas
	Deborah Bell-Pedersen
Head of Department,	Emil Straube

August 2019

Major Subject: Mathematics

ABSTRACT

Biological clocks generate rhythms with periods from seconds to months in many organisms and control many processes that are critical to the survival of the organism. Many rhythms in biology are the result of rhythms in the mRNA and protein abundances at the cellular level that are then synchronized within the organism. To better understand biological clocks, mathematical and computational modeling is crucial as these tools provide directions for experimental procedures, test hypotheses within these models, and corroborate new biological revelations. Therefore, we apply novel mathematical and computational techniques to obtain insights into biological clocks at the molecular level.

First, we generalize and analyze an ODE model of the repressilator with an arbitrary number of genes. Previous models of the repressilator were derived from assumptions that are biologically restrictive. These previous models assume first-order transcription, translation, and degradation, with rates equivalent among genes, mRNAs, and proteins, respectively. This assumption, however, is not consistent with current biological knowledge. Accordingly, we propose a new repressilator model allowing for differing transcription, translation, and degradation terms. We show that, under conditions on these new functions, there is still a unique steady state when an odd number of genes are in the network. We also show that, with an odd number of genes, either the model converges to the steady state or to a periodic orbit. Finally, we give a counterexample to a recent conjecture proposed by Tyler *et al*. Taken together, our results advance the theoretical study of cyclic gene repression by generalizing the current repressilator models.

Next, we derive a new transcription-rate function that arises from more reasonable biological assumptions than the traditionally used transcription-rate function. Furthermore, we analyze the qualitative differences in the period, amplitude, and phase of a model with our new transcription-rate function and a model with the old transcription-rate function. Our numerical simulations reveal drastic differences in the period, amplitude, and phase of the protein waveforms between the two models.

Finally, we present novel algorithms to address the parameter estimation of the repressilator. Parameter estimation of models of biological oscillators is inherently challenging due to the nonlinearity present in the system. We illustrate two specific challenges by attempting to fit the parameters of the repressilator using traditional techniques. We then revisit two standard parameter estimation procedures and show how they fail to generate accurate parameter estimates. Next, we present three novel algorithms, two that take as input time-course data of the repressilator and output parameter estimates. The third algorithm takes as input time-course data of the repressilator along with a specific parameter estimate and outputs the period of the model solution with that parameter estimate as well as model solution values at the time points of the data. Ultimately, we show that our new procedures are more accurate than the two standards in the field of computational biology and a commonly used global optimization procedure, the particle swarm algorithm.

DEDICATION

To my parents, Amanda and Dan, for affording me every opportunity.

To my wife, Meredith, for loving and supporting me in everything.

To my children, Marie and James, for bringing me immeasurable joy.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my two advisors: Anne Shiu and Jay Walton. They taught me how to conduct mathematical research, how to communicate effectively my results in manuscripts and presentations, how to collaborate with other scientists, and how to balance a life between work and family. I am blessed to have had two such distinguished, attentive, and patient advisors.

Second, I would like to thank my two other committee members, Harold Boas from the Department of Mathematics and Deborah Bell-Pedersen from the Department of Biology. Harold Boas was quick to provide much comic relief in the stressful moments of my academic journey, which I always appreciated. I am grateful to Deborah Bell-Pedersen for providing invaluable insights into the biology of biological clocks that made my work significantly better.

Next, I would like to thank the graduate program at Texas A&M, specifically Peter Howard, the Director of Graduate Studies, and Monique Stewart. Peter Howard was always willing to listen and offer sage advice. Monique Stewart patiently endured my serial procrastination while endlessly confirming that I had the proper paperwork submitted and the necessary requirements satisfied.

In addition, I am grateful to the Texas A&M High Performance Research Computing center for allowing me to use the computing resources to conduct portions of this research.

Also, I would like to thank my fellow graduate students in the math department (in particular, Joe Torres, Mahishanka Withanachchi, Brian Hunter, Burak Hatinoglu, Alexander Ruys de Perez, Nida Obatake, Ola Sobieska-Snyder, Ayomikun Adeniran, Taylor Brysiewicz, and David Sykes). It has been an honor working alongside you, learning from you, and growing into mathematicians with you. I thoroughly enjoyed our extended homework sessions, outdoor game days, game nights, cookie breaks, writing groups, GSO seminars, and the select fluid dynamics that I attended. I am excited to follow all of your careers and to witness all of your successes and joys in mathematics and in life.

I must also thank all of the friends that I have made outside of the math department here in Bryan/College Station. Nerd frisbee on Sunday evenings was always a much-needed diversion from the rigors of mathematical research. All of my friends at St. Mary's were instrumental in facilitating my growth as a person, friend, husband, and father. I will always cherish the friends I made through singing and playing with the 8 am choir.

Furthermore, I would like to thank my parents, Amanda and Dan. It truly is because of them that I have done anything of circumstance in my life. They provided a nurturing home full of fun, love, support, intellectual stimulation, and pride in their children. To them, I am eternally grateful. I am also grateful to my younger sister, Abigail. She was the best partner in crime growing up, and she continues to inspire me to pursue my passions in life. I am also blessed to have another set of parents in my in-laws, Marian and Kevin. I am grateful that they have embraced me as their own and have loved and supported me in this journey.

Finally, I would like to thank my wife, Meredith. She is truly the reason that this document exists. She believed in me even when I didn't. She pushed me when I couldn't find the motivation within myself. She supported me emotionally and financially. She cared for both of our children so that I would have no distractions in my last year. All the while, she loved me and cared for me. I cannot imagine having a better wife at my side.

And speaking of our children, Marie and James, there is nothing better in life than walking into your home in the evening to two smiling children that have waited patiently all day for your arrival. When they raced (either running or crawling) to see me, all stress in life instantly evaporated.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Dr. Jay Walton (co-advisor), Dr. Anne Shiu (co-advisor), and Dr. Harold Boas of the Department of Mathematics and Dr. Bell-Pedersen of the Department of Biology.

The student independently completed all work for the dissertation.

Funding Sources

Graduate study was supported by a teaching assistantship from Texas A&M University as well as partial support from the NSF (DMS-1752672).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 From Biological Clocks to Circadian Rhythms

Across all Kingdoms in biology, biological clocks generate rhythms with periods from seconds to months. These clocks control essential processes, which are critical for organism survival, such as cell division [7], embryogenesis [8], DNA damage repair and metabolism [9, 10, 11, 12], and the annual migration patterns of birds [13]. Arguably, the most important biological clock, however, is the *circadian clock* which coordinates many processes in the 24 hour day. Perhaps most notably, the clock regulates the sleep-wake cycle in mammals and other organisms by controlling the secretion of melatonin starting in the evening and ending in the early morning. Other key clock outputs include rates of metabolism, cognitive function, blood pressure, basal body temperature, and hormonal secretion (Figure 1.1).

Because the circadian clock controls essential physiological processes, a disruption in the circadian clock leads to many pathological conditions. The most common example is jet lag, the annoying effect of traveling across time zones. Feelings of exhaustion and fogginess are the result of the circadian clock lagging in synchronizing to the new environmental light/dark cycle. More serious disorders can be caused by defects in one's circadian clock. For example, shift work, which causes a desynchronization between processes controlled by the worker's circadian rhythm and the time the worker is active, has been labeled a class 2A carcinogen [14]. The designation 2A means limited evidence of carcinogenicity in humans but sufficient evidence of carcinogenicity in experimental animals. Other 2A carcinogens include red meat, repeated exposure to petroleum refineries, and anabolic steroids.

To better understand biological clocks, mathematical models are necessary because such models identify key properties of the clock quickly and with little expense. Biological experiments take years to complete at a high price tag, but mathematicians can help by using models to provide directions for experimental procedures, to test hypotheses within these models, and to corroborate

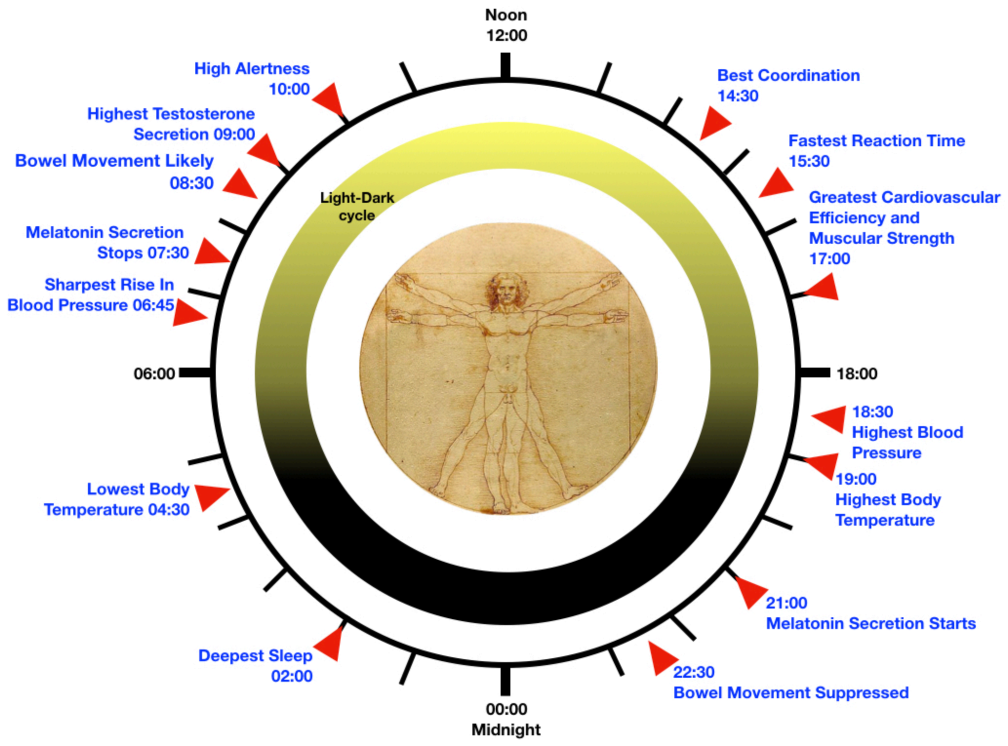


Figure 1.1: Human circadian rhythm. Adapted from a figure in *The Body Clock Guide to Better Health* [1].

new biological revelations. From the molecular to the macro level, many mathematical models provide insights into the structure and dynamics of biological clocks.

One such example of a model providing insight into the molecular structure of the clock is the discovery of interlocked feedback loops. Researchers knew from the 1960s that negative feedback loops were essential for oscillations of the various cellular components. Since mathematical models provide the advantage to test many possible regulatory networks, in the early 2000s, modelers began to compare network structures with and without positive feedback loops [10, 15, 16]. A model by Becker-Weimann *et al*. revealed that components of a positive feedback loop are more sensitive to parameter variations than components of a negative feedback loop. They also found that these components have more variable phases and amplitudes than the negative components, which allows for control of output pathways without disturbing core clock oscillations [17]. Multiple feedback loops also provide many sources of oscillations supplying necessary redundancy to protect against mutations in genetic components of the clock [18].

In this dissertation, we present a rigorous mathematical and computational analysis of a specific biological clock called the *repressilator* [5]. The repressilator was introduced as a synthetic gene network that, within *E. coli* cells and across generations, exhibited oscillations in its proteins [5]. The network was later identified as the core molecular circadian mechanism in *Arabidopsis thaliana* [3, 19]. The repressilator provides a foundation for studying the molecular mechanisms of biological clocks and the mechanism of repression on the transcription of genes.

1.2 Overview of the Main Results

In Section 2, we motivate and introduce a new mathematical model of the repressilator (System (2.7) in Section 2). The new system generalizes a previous model by Müller *et al.* in [20] by removing two biologically restrictive assumptions. We prove that many results of previous repressilator models extend to our generalized model. First, with an odd number of genes, the system has a unique steady state, called the *central steady state* (Theorem 2.3.3 in Section 2.3.1.1), and the system converges to that steady state or produces limit-cycle oscillations (Theorem 2.3.20 in Section 2.3.4). Next, we prove a necessary and sufficient condition for the stability of any steady state in the case of an even number of genes (Theorem 2.3.13 in Section 2.3.2.1). Finally, we give a counterexample to Conjecture 1 proposed by Tyler *et al.* in [21] (Section 2.3.3). With each result, we discuss the biological implications of the mathematical analysis.

Next, in Section 3, we derive a new function that models the transcription rate in the presence of a repressor protein (Eqn. (3.14) in Section 3.2.2). The new function arises under more reasonable assumptions than the traditionally used Hill function (see Remark 2.3.1 in Section 2). We prove that the new function satisfies the assumptions placed on transcription-rate functions in Section 2.3 in Section 2. Also, we investigate the quantitative and qualitative differences between model solutions of the repressilator with the Hill function and model solutions of the repressilator with the new transcription-rate function. We show that there are drastic differences in the period, amplitude, and phase between the two model solutions. Finally, we prove partial results towards a conjecture that the amplitude of protein abundance of a model with the old transcription-rate function is greater than the amplitude of protein abundance of a model with our new transcription-rate function, all other parameters and functions being equal.

In Section 4, we perform a comprehensive investigation into new algorithms to estimate the parameters of the repressilator system. First, we discuss two challenges to parameter estimation of the repressilator (Section 4.3) and show how previous parameter estimation procedures fail to address both at the same time (Section 4.4). Then, we analyze potential quantities to incorporate into an identifiability test to produce accurate parameter estimates. Subsequently, we outline two new algorithms that take as input time-course data of protein abundance profiles and output a parameter estimate of the repressilator system (Section 4.6). We investigate the effectiveness of the new algorithms on simulated data sets (Section 4.7). Also, we run both algorithms on 100 data sets to compare the accuracy of each and, with the empirical evidence, conclude that the second is more accurate than the first. We finish by using simulated data sets to compare the second algorithm to standard algorithms in computational biology (Section 4.8.1). Our analysis reveals that our new algorithm is more accurate and faster than existing algorithms.

Finally, in Section 5, we review all three studies and highlight the importance of our mathematical and computational analysis to answering questions in the field of biological clocks. Moreover, we propose future directions that will expand the studies.

2. A SYNTHETIC INTRACELLULAR REGULATORY NETWORK THAT EXHIBITS OSCILLATIONS[1]

2.1 Introduction

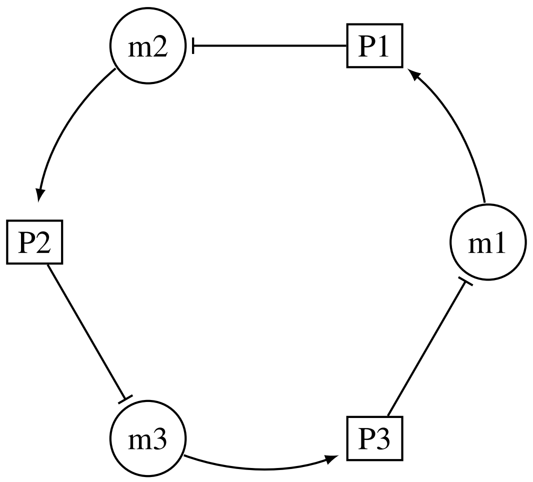


Figure 2.1: The repressilator network with three genes and their respective products [2]. The m's denote mRNA while the P's denote proteins. The product of gene-1 represses transcription of gene-2; the product of gene-2 represses transcription of gene-3; the product of gene-3 represses transcription of gene-1. The figure is used with permission from [21].

The seminal work of Elowitz and Leibler is a consequence of the paradigm shift in experimental biology occurring late in the 20th century when the field shifted from the traditional quantitative analysis approach to the newly emerging field of synthetic biology. Even for simple intracellular systems, *design principles*–rules that characterize some biological feature exhibited by a class of systems [22]–are difficult to identify using traditional quantitative analytic techniques. Synthetic biology complements quantitative analysis by designing and constructing synthetic networks that implement a particular function. The insights arising from synthetic biology are twofold: engineering of new cellular behaviors and improved functional understanding of naturally occurring

[1]Based on Sections 1 and 2 of "Revisiting a synthetic intracellular regulatory network that exhibits oscillations," by J. Tyler, A. Shiu, and J. Walton, 2019. *J MATH BIOL*, 78:2341-2368. Used with permission.

networks.

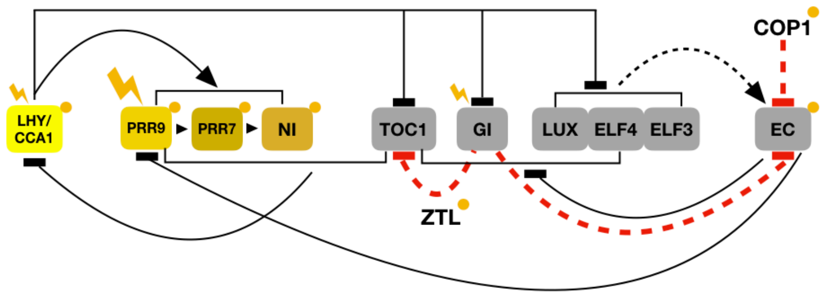
In their work designing and implementing the repressilator, Elowitz and Leibler aimed to better understand naturally occurring networks that exhibit oscillations. In particular, an important question in the field of biological clocks is: what cellular networks are sufficient for sustained oscillations within the cell? Using the repressilator, they showed that, in fact, a circuit of pure repression of three genes is sufficient to exhibit sustained oscillations within the cell and across generations. This important insight has led to recent work in identifying organisms whose circadian clocks–or other biological clocks–are driven by repressilator circuits [3, 19].

2.1.1 The Repressilator as an Architecture for Circadian Clocks

Currently, there are three leading classes of circadian models: *Hill-type repression*, *protein sequestration*, and the repressilator [23]. Hill-type repression models are derived from the assumption that the repressor facilitates a configuration change–e.g. through several phosphorylations, catalyzed by the repressor, of the activator–to inhibit the binding of the activator to the promoter. In contrast, for the protein sequestration mechanism, the repressor tightly binds to the activator to form a 1:1 stoichiometric complex, rendering the activator inactive [24, 25, 26, 27]. Protein sequestration is currently thought to be the mechanism of action for the mammalian circadian clock and others such as the cyanobacteria clock [28, 29]. There is a debate in the field about the mechanism of other circadian clocks, such as the *Neurospora crassa* circadian clock, which is hypothesized to utilize a Hill-type repression mechanism [30, 31, 32, 33, 34].

While the Hill-type repression and protein sequestration models seem to cover the structure of circadian clocks in animals and prokaryotes, the third architecture, the repressilator, is thought to be the core mechanism of action for the plant *Arabidopsis thaliana* circadian clock (Figure 2.2) [3, 19]. The morning transcription factors CCA1 (Circadian Clock Associated 1) and LHY (Late Elongated Hypocotyl) repress the formation of the evening complex, EC. The EC consists of the proteins LUX, ELF3, and ELF4, which all peak in the evening and form a tripartite protein complex (Figure 2.2A). The EC then inhibits the production of pseudoresponse regulators 5, 7, and 9 (PRR5, PRR7, and PRR9) in the late evening to allow for the production of the LHY/CCA1

A



B

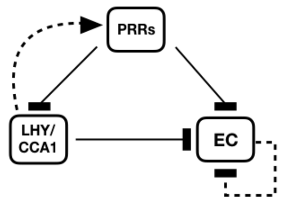


Figure 2.2: (A) Outline of the *Arabidopsis thaliana* circadian clock. Components in yellow comprise the daytime elements and grey components, the nighttime elements. Solid lines denote transcriptional regulation. Red dashed lines denote post-translational regulation. (B) Simplified schematic of the *Arabidopsis* circadian clock to highlight the repressilator architecture present within the system. Adapted from Figures 1 and 8 in [3].

complex to peak in the morning. Finally, the pseudoresponse regulators (PRRs) peak in the daytime and repress the production of the LHY/CCA1 complex (Figure 2.2A). These three components form the core negative feedback loop, which has the same structure as a repressilator (Figure 2.2B) [3, 19].

Because the repressilator is important in the field of synthetic biology and has been recognized as the core mechanism of the circadian clock in plants, we seek a rigorous mathematical understanding of this system. Accordingly, the work of this section addresses a generalized model of the repressilator that is more faithful to biology. In particular, we allow for more types of functions to model the processes of transcription, translation, and degradation than previous mathematical models allow. The purpose of the work is twofold: a better functional understanding of the plant circadian clock and the process of repression within a gene network in general and insights into design principles of new repressilator circuits with implications into synthesizing new biotechnologies.

2.1.2 Section Organization

The section is organized as follows. In Section 2.2, we introduce necessary background material including previous mathematical models of the repressilator (Sections 2.2.1 and 2.2.2) and results from the theory of ODEs and dynamical systems (Sections 2.2.3 and 2.2.4). In Section 2.3, we motivate and introduce our generalized model of the repressilator. We prove that many of the results of previous repressilator models extend to our generalized model. First, with an odd number of genes, the system has a unique steady state, called the *central steady state* (Section 2.3.1.1), and the system converges to that steady state or produces limit-cycle oscillations (Section 2.3.4). Next, we prove a necessary and sufficient condition for the stability of any steady state in the case of an even number of genes (Section 2.3.2.1). We also discuss what the condition means biologically. Finally, we end with a discussion in Section 2.4.

2.2 Background

Here, we present the relevant background material for the subsequent model construction and analysis. In particular, we review the original repressilator model given by Elowitz and Leibler in [5]. Next, we recall the mathematical model introduced by Müller *et al.* and a few key results regarding steady states, stability, and the asymptotic behavior [20]. Finally, our work requires the application of the Routh-Hurwitz Theorem and a key result from monotone systems theory, so we review the two briefly.

2.2.1 Elowitz and Leibler Mathematical Model

In addition to presenting experimental results, Elowitz and Leibler also introduced a mathematical model to describe the dynamics of the three gene repressilator (Figure 2.1). Their mathematical model is given by the following system of 6 ODEs [5]:

$$\begin{aligned}
\dot{r}_1 &= \frac{\alpha}{1+p_3^n} + \alpha_0 - r_1, & \dot{p}_1 &= -\beta(p_1 - r_1) \\
\dot{r}_2 &= \frac{\alpha}{1+p_1^n} + \alpha_0 - r_2, & \dot{p}_2 &= -\beta(p_2 - r_2) \\
\dot{r}_3 &= \frac{\alpha}{1+p_2^n} + \alpha_0 - r_3, & \dot{p}_3 &= -\beta(p_3 - r_3).
\end{aligned} \tag{EL}$$

Here, the state m_i corresponds to the i-th mRNA; the state p_i corresponds to the i-th protein, which is translated from m_i. The parameter α is the transcription rate shared by all three mRNAs. The parameter α_0 refers to the *leakiness* of the promoter–the rate of transcription that occurs in saturated amounts of repressor. The parameter β is the ratio of the protein decay rate to the mRNA decay rate. Finally, the parameter n is the *Hill coefficient*.

For system (EL), Elowitz and Leibler addressed certain key questions in dynamical systems and mathematical modeling of biological systems, the first of which is:

Question 2.2.1. What are the possible steady states of the system?

The structure of system (EL) guarantees that there is a unique steady state characterized as the vector $x^* = (\bar{p}, \ldots, \bar{p})$. Here, the quantity $\bar{p}$ solves the equation

$$\bar{p} = \frac{\alpha}{(1+\bar{p}^n)} + \alpha_0 \quad [5].$$

With the unique steady state x^*, the natural next question is

Question 2.2.2. Under what conditions is the steady state stable?

Again, the structure of system (EL) simplifies the stability analysis. The unique steady state x^* is stable when

$$\frac{(\beta+1)^2}{\beta} < \frac{3X^2}{4+2X},$$

where

$$X = \frac{-\alpha n \bar{p}^{n-1}}{(1+\bar{p}^n)^2} \quad [5].$$

Finally, the authors addressed:

Question 2.2.3. What possible asymptotic behavior can the system exhibit?

Through numerical simulations, Elowitz and Leibler showed that the system can stabilize or oscillate. Also, they plotted stable and unstable domains when letting the parameters α, β, n, and α_0 vary. They showed that the unstable domain increases with increasing Hill coefficient while an increase in the leakiness causes the unstable domain to shrink [5]. In subsequent sections, we investigate more deeply how the repressilator is affected by changes in the Hill coefficient as well as the biological interpretation of the Hill coefficient.

2.2.2 Müller *et al*. Mathematical Model

To incorporate an arbitrary number of genes and to make the mathematical analysis more rigorous, in 2006, Müller *et al.* generalized system (EL) [20]. Specifically, Müller *et al.* analyzed two systems of ODEs that describe the dynamics of a repressilator with an arbitrary number of genes. One system assumed that, in saturated amounts of repressors, transcription occurs at a very low rate. Muller *et al.* called this system *RepLeaky* and addressed Questions 2.2.1-2.2.3, among others. In particular, they proved results about the number of steady states, the stability of those steady states, and the limiting dynamics [20]. Here, the RepLeaky system is the starting point for our generalized repressilator model.

The RepLeaky system of Müller *et al.* arose from five key assumptions [20]:

(a) Genes are present in constant amounts.

(b) When a protein binds to a regulatory element of a gene, it either enhances or inhibits transcription. Also, binding reactions are in equilibrium.

(c) Transcription and translation operate under saturated conditions.

(d) Both mRNAs and free proteins are degraded by first-order reactions.

(e) Transcription, translation, and degradation rates are the same among genes, mRNAs, and proteins, respectively.

Müller *et al.*'s RepLeaky model [20], which arises from a generalization of Figure 2.1 to n genes, is given by the following system of $2n$ ODEs where n denotes the number of genes:

$$\begin{aligned} \dot{r}_i &= \alpha f(p_{i-1}) - r_i, \\ \dot{p}_i &= \beta r_i - \beta p_i, \end{aligned} \tag{2.1}$$

for $i = 1, ..., n$. Here, p_i denotes the concentration of protein-i, where i is viewed mod n, and r_i denotes the mRNA concentration. The parameter β is the ratio of protein degradation to mRNA degradation, and the parameter α is the transcription rate. The function $f(x)$ models the repression of gene-i transcription resulting from repressor protein-$(i-1)$ binding to the promoter (see Figure 2.1):

$$f(x) = \frac{1-\delta}{1+x^h} + \delta,$$

where the parameter δ is the ratio of repressed to unrepressed transcription [20]. Synthesis of protein-i occurs by translation of mRNA-i and is proportional to the mRNA-i concentration. Degradation of each species is modeled by a first-order term proportional to its own concentration.

Again, the authors are concerned with the questions of the number of steady states and conditions for stability of these steady states. They show that, when system (2.1) has an odd number of genes, there is a unique steady state, denoted E_C for central steady state. When the system has an odd number of genes, the unique steady state E_C is stable if and only if

$$\frac{\beta}{(1+\beta)^2} < \frac{1 - S_c \cos(\pi/n)}{S_c^2 \sin^2(\pi/n)}, \tag{2.2}$$

where

$$S_c = -\alpha f'(E_C). \tag{2.3}$$

For system (2.1) with an even number of genes, the dynamics differ. When system (2.1) has an even number of genes, it has exactly one or three steady states. The central steady state is always a steady state. The proposition below gives conditions for stability of the steady states of system (2.1) with an even number of genes.

Proposition 2.2.4. [20, Theorem 1] If $S_C = -\alpha f'(E_C) < 1$, then E_C is globally asymptotically stable and is the only steady state. If $S_C > 1$, then E_C is unstable and there are two more steady states, E_{odd} and E_{even}. Moreover, if $S_C > 1$, then almost all orbits converge to E_{odd} or E_{even}.

Finally, the authors address the asymptotic behavior of the system with an odd number of genes in the proposition below.

Proposition 2.2.5. [20, Theorem 2] For n odd, system (2.1) has the following property: (i) Every orbit converges to E_C or to a periodic orbit. (ii) If E_C is unstable, then there exists a periodic attractor.

2.2.3 Routh-Hurwitz Criterion

Throughout the section, we will refer to the Routh-Hurwitz criterion, so we review it briefly. Consider a univariate polynomial:

$$p(x) = a_n + a_{n-1}x + a_{n-2}x^2 + ... + a_0x^n. \tag{2.4}$$

Definition 2.2.6. For $k = 1, \dots, n$, the k^{th} **Hurwitz matrix** of p as in (2.4) is the $k \times k$ matrix $H_k = [h_{ij}]_{i,j=1}^k$, defined by $h_{ij} = a_{2i-j}$, where a_{2i-j} is defined as 0 if $2i - j < 0$ or $2i - j > n$.

For example, the fourth Hurwitz matrix of $p(x) = a_4 + a_3x + a_2x^2 + a_1x^3 + a_0x^4$ is:

$$H_4 = \begin{bmatrix} a_1 & a_0 & 0 & 0 \\ a_3 & a_2 & a_1 & a_0 \\ 0 & a_4 & a_3 & a_2 \\ 0 & 0 & 0 & a_4 \end{bmatrix}.$$

Following the notation in [35], we write $D_i = \det(H_i)$.

Proposition 2.2.7. [36, Routh-Hurwitz Criterion] Consider a polynomial p as in (2.4). Every root of p has negative real part if and only if the determinants of all Hurwitz matrices (Definition 2.2.6) are positive, i.e.,

$$D_i > 0, \quad i = 1, 2, ..., n.$$

Recall that the stability of a steady state is characterized by negative real parts of the roots of the characteristic polynomial of the Jacobian. Thus, in Section 2.3.2, we apply Proposition 2.2.7 to the characteristic polynomial to obtain conditions for the stability of a steady state.

Remark 2.2.8. In his seminal work on Mathematical Biology, Murray stated of the Routh-Hurwitz Criterion [37]: "Frankly, it is hard to imagine anyone actually using the conditions for polynomials of order five or more." Nevertheless, the Routh-Hurwitz Criterion is still a widely used technique to analyze the stability of steady states for systems with many more than five species. The advancement of computational software since Murray wrote his textbook has made it easier to implement the criterion. For the repressilator, the criterion reduces nicely as we will see in Section 2.3.2.

2.2.4 Monotone Systems Theory

Finally, we review results of Mallet-Paret and Smith from their study of monotone cyclic feedback systems of ODEs. First, we recall that a *monotone cyclic feedback system* in n coordinate variables is a system of the form

$$\dot{x}_i = f_i(x_{i-1}, x_i), \quad i = 1, \ldots, n, \tag{2.5}$$

where x_0 is interpreted as x_n [38]. Note that both systems (EL) and (2.1) are of the form in (2.5) when the states are reordered as r_1, p_1, ..., r_n, and p_n.

A key assumption of Eqn. (2.5) is that the variable x_{i-1} forces $\dot{x}_i$ monotonically. Thus, there is some $\delta_i = \{-1, 1\}$ such that

$$\delta_i \frac{\partial f_i(x_i, x_{i-1})}{\partial x_{i-1}} > 0 \quad \text{for all} \quad 1 \leq i \leq n.$$

If $\delta_i = -1$, then the state x_{i-1} inhibits the growth of state x_i. Likewise, if $\delta_i = 1$, then the state x_{i-1} promotes the growth of x_i. For systems (EL) and (2.1), the quantity δ_i corresponding to the mRNA growth rate is -1 because the previous state, p_{i-1}, inhibits the growth of r_i. However, the quantity δ_i corresponding to a protein is 1 because the previous state, r_i, promotes the production of the protein.

Given the individual δ_i terms, Mallet-Paret and Smith defined the product

$$\Delta = \delta_1 \cdots \delta_n, \tag{2.6}$$

which characterizes the system as a *negative feedback loop* if $\Delta = -1$ and a *positive feedback loop* if $\Delta = 1$ [38]. The repressilator system (2.1) is a negative feedback loop when n is odd and a positive feedback loop when n is even. This disparity drives the variation in dynamics between our repressilator model with an odd number of genes and ours with an even number of genes.

Moreover, Mallet-Paret and Smith showed that the omega limit set of solutions to monotone cyclic feedback systems can be embedded in $\mathbb{R}^2$, and in fact, must be of the form: either a single equilibrium, a single, nonconstant periodic solution, or a set of equilibria together with homoclinic and heteroclinic orbits connecting these equilibria [38]. These results are summarized in the proposition cited below.

Proposition 2.2.9. [38, Main Theorem] (a) Let $x(t)$ be a solution of the monotone cyclic feedback system (2.5) through $x(0) = x_0$, and suppose the forward orbit $\gamma^+(x_0)$ is bounded. Then the omega limit set $\omega(x_0)$ is one of the following:

(i) an equilibrium

(ii) a nonconstant periodic orbit

(iii) a set $E \cup H$ where each $y_0 \in E$ is an equilibrium at which $\Delta \det(Df(y_0)) \geq 0$, with Δ as in (2.6), and where H is a set of orbits homoclinic to/heteroclinic between points of E; that is, if $y_0 \in H$, then $\alpha(y_0) = \{z_0\}$ and $\omega(y_0) = \{w_0\}$ for some $z_0, w_0 \in E$. There, moreover, exists an integer k_∞, which is odd if $\Delta = -1$ and even if $\Delta = 1$, such that for each $z_0 \in E$ the matrix $Df(z_0)$ has either $k_\infty - 1$ or k_∞ eigenvalues γ satisfying $Re(\gamma) > 0$.

2.3 Generalized Repressilator Model

Here, we introduce the generalized repressilator model which allows for more general functions to model the transcription, translation, and degradation processes. We motivate our generalization by considering specific transcription, translation, and degradation mechanisms found in the circadian biology literature. Our generalized model provides key insights into the processes of transcription, translation, and degradation as well as insights into the design of new repressilator circuits.

First, recall from Section 2.2.2 that the Müller *et al*. model was based on five assumptions [20]:

(a) Genes are present in constant amounts.

(b) When a protein binds to a regulatory element of a gene, it either enhances or inhibits transcription. Also, binding reactions are in equilibrium.

(c) Transcription and translation operate under saturated conditions.

(d) Both mRNAs and free proteins are degraded by first-order reactions.

(e) Transcription, translation, and degradation rates are the same among genes, mRNAs, and proteins, respectively.

Two of these assumptions are too biologically restrictive, so we generalize the model by removing them. Consider, for example, the translation process. In eukaryotic cells, mRNAs must be spliced correctly before they exit the nucleus and then be translated [39]. Similarly, since transcription depends on the uncoiling of DNA due to different locations of genes on histones [40], transcription rates should be allowed to vary across genes. Finally, ubiquitization, which facilitates degradation, also differs extensively among proteins [41]. Thus, to be more faithful to the biology, we remove assumption (e).

Next, we consider assumption (d). Recently, Page and Perez-Carrasco analyzed the repressilator after allowing for differing degradation rates among the proteins [42]. Here, we argue for a further generalization. In the context of the degradation pathway of a core clock component of the Neurospora circadian clock, phosphorylation of the FREQUENCY (FRQ) protein initiates its own degradation. This process occurs through the ubiquitin-proteasome pathway, which is a Michaelis-Menten pathway [43]. Modeling the rate of FRQ degradation as proportional to its concentration is therefore not appropriate. Thus, for our repressilator model, we remove assumption (d) to allow for more general functions than first-order terms.

Our **generalized n-gene repressilator system**, which generalizes (2.1), is given by the following system of ODEs:

$$\begin{aligned}
\dot{r}_1 &= a_1(p_n) - d_{r_1}(r_1), \\
&\vdots \\
\dot{r}_n &= a_n(p_{n-1}) - d_{r_n}(r_n), \\
\dot{p}_1 &= k_1(r_1) - d_{p_1}(p_1), \\
&\vdots \\
\dot{p}_n &= k_n(r_n) - d_{p_n}(p_n).
\end{aligned} \tag{2.7}$$

Here, for the i-th gene, r_i is the concentration of mRNA-i, and p_i is the concentration of protein-i. Each equation in the system has a synthesis term and a degradation term. One synthesis term is the function $a_i(p_{i-1})$, called the *transcription-rate* function of gene-i in terms of protein-($i-1$). The degradation term for mRNA-i is the *degradation-rate* function $d_{r_i}(r_i)$, which is a function of its own concentration. The function $k_i(r_i)$ is the *translation-rate* function describing the synthesis of protein-i in terms of mRNA-i. Finally, the *degradation-rate* function $d_{p_i}(p_i)$ models the degradation of protein-i as a function of its own concentration.

The 3-gene version of system (2.7) reflects Figure 2.1. The **m1** node denotes mRNA-1 which translates, according to the function $k_1(r_1)$, to protein-1, **P1**. This protein then represses the synthesis of the second mRNA, which is described by the transcription-rate function $a_2(p_1)$.

Next, we give conditions on the transcription-rate, degradation-rate, and translation-rate functions that we assume for the results below. These assumptions are rooted in the biology of the specific process they model. For the transcription-rate functions, we begin with the biological assumptions.

(B1) Transcription rates vary smoothly in the amount of repressor present.

(B2) Transcription rates are always nonnegative.

(B3) Transcription rates decrease with increased repressor present.

(B4) Transcription rates are positive when no repressor is present.

These biological assumptions translate into the following mathematical assumptions on the transcription-rate function $a_i(x)$:

(A1) $a_i(x) \in C^1[\mathbb{R}_{\geq 0}]$.

(A2) $a_i(\mathbb{R}_{\geq 0}) \subset \mathbb{R}_{\geq 0}$.

(A3) $a_i(x)$ is strictly decreasing on $\mathbb{R}_{\geq 0}$.

(A4) $a_i(0) > 0$.

Remark 2.3.1. The canonical transcription-rate function, introduced by Goodwin in 1965 [44], is called the *Hill function*, which is $a_i(p) = \frac{k_i^S}{1+p^h}$ [45]. The exponent h is the Hill coefficient, which we saw earlier in system (EL). It is easily seen that this function satisfies the assumptions (A1)-(A4). In Section 3, we derive a new transcription-rate function from more reasonable biological assumptions and compare the dynamics of a model with the Hill function and a model incorporating our new transcription-rate function.

Next, we provide biological assumptions for degradation-rate and translation-rate functions.

(B1) Degradation and translation rates vary smoothly in the protein or mRNA concentration.

(B2) Degradation and translation rates occur only when the protein or mRNA is present.

(B3) Degradation and translation rates increase as protein or mRNA concentrations increase.

These assumptions give rise to the following mathematical assumptions on the degradation-rate and translation-rate functions $d_{p_i}(x)$, $d_{r_i}(x)$, and $k_i(x)$.

(D1) $d(x), k(x) \in C^1[\mathbb{R}_{\geq 0}]$.

(D2) $d(0) = k(0) = 0$.

(D3) $d(x), k(x)$ are strictly increasing on $\mathbb{R}_{>0}$.

Notice immediately that degradation-rate and translation-rate functions satisfying (D1)-(D3) are invertible on their ranges. This will be important in the analysis below.

For the remainder of the section, when considering our repressilator system (2.7), we assume that the functions $a_i(p_{i-1})$ satisfy (A1)-(A4), and the functions $d_{p_i}(p_i)$, $d_{r_i}(r_i)$, and $k_i(r_i)$ satisfy (D1)-(D3).

2.3.1 Steady States

For system (2.1), Müller *et al.* proved the existence of a unique steady state, labeled E_C for *central steady state*, in the odd-n case and also showed that E_C exists in the even-n case [20]. When we allow general transcription-rate and degradation-rate functions in system (2.7), however, we are not always guaranteed a steady state. Consider the following example.

Example 2.3.2. Consider the following 2-gene version of the repressilator system (2.7):

$$\begin{aligned}
\dot{r}_1 &= 2\pi - \arctan(p_2) - r_1 \\
\dot{r}_2 &= 2\pi - \arctan(p_1) - r_2 \\
\dot{p}_1 &= r_1 - \arctan(p_1) \\
\dot{p}_2 &= r_2 - \arctan(p_2).
\end{aligned} \tag{2.8}$$

It is straightforward to check that the assumptions (A1)-(A4) and (D1)-(D3) hold for the corresponding functions $a_i = 2\pi - \arctan(p_{i-1})$, $d_{r_i} = r_i$, and $d_{p_i} = \arctan(p_i)$. We set the equations in (2.8) to zero to solve for the steady states, giving

$$2\pi - \arctan(p_2) = \arctan(p_1) \tag{2.9}$$

$$2\pi - \arctan(p_1) = \arctan(p_2). \tag{2.10}$$

However, Eqns. (2.9) and (2.10) have no positive, real solution. Therefore, system (2.8) has no steady state. The same is true if we augment system (2.3.2) to three genes using the same functions for the mRNA and protein, respectively.

What went wrong in this example? The degradation-rate function d_{p_i} and the transcription-rate function a_i each had a horizontal asymptote that prevented intersection of their respective graphs in $\mathbb{R}^2_+$. This lack of intersection precluded the existence of a steady state. So, to prove when steady states exist, we must introduce more assumptions.

Notice that assumptions (A2) and (A3) imply:

$$\alpha_i := \lim_{x\to\infty} a_i(x) < \infty \quad \text{and} \quad \lim_{x\to\infty} a_i'(x) = 0.$$

This parameter α_i corresponds to the *leakiness* of the promoter of gene-i [20]. If $\alpha_i > 0$, then even in saturated amounts of repressor, gene-i will still be transcribed at a positive rate, whereas $\alpha_i = 0$ implies that in saturated amounts of repressor, gene-i will not be transcribed. We introduce a new assumption on the transcription-rate function $a_i(p)$.

(A5) $\alpha_i = 0$ for all $i = 1, ..., n$.

Even if the leakiness α_i is nonzero, we avoid the problem highlighted in Example 2.3.2 by introducing an assumption on the relationship among the transcription-rate and degradation-rate functions. Let us define

$$\delta_i^R := \lim_{x\to\infty} d_{r_i}(x) \quad \text{and} \quad \delta_i^P := \lim_{x\to\infty} d_{p_i}(x).$$

We allow for δ_i^R and δ_i^P to be infinite. The δ_i^P's and δ_i^R's correspond to the maximum possible degradation rate for protein-i and mRNA-i, respectively. To avoid the problem in Example 2.3.2, we introduce the following relationship among δ_i^R, δ_i^P, and a_i:

(A6) $\delta_i^P > k_i(d_{r_i}^{-1}(a_i(0)))$ and $\delta_i^R > a_i(0)$, for all $i = 1, ..., n$.

Below, by using combinations of the above assumptions and others, we prove conditions under which E_C exists, first with an odd number of genes, and then with an even number.

2.3.1.1 Odd-n Case

For system (2.1), Müller *et al.* showed that the system has a unique steady state [20]. We prove that this property extends to system (2.7).

Theorem 2.3.3. For n odd, if system (2.7) satisfies (A6), then system (2.7) has a unique steady state in $\mathbb{R}_+^{2n}$.

Proof. First, we set the equations in system (2.7) to zero:

$$0 = \dot{r}_i = a_i(p_{i-1}) - d_{r_i}(r_i) \implies d_{r_i}(r_i) = a_i(p_{i-1}) \tag{2.11}$$

$$0 = \dot{p}_i = k_i(r_i) - d_{p_i}(p_i) \implies d_{p_i}(p_i) = k_i(r_i). \tag{2.12}$$

From Eqns. (2.11) and (2.12), it is easy to check that finding steady states reduces to finding solutions to the system

$$p_i = d_{p_i}^{-1} \circ k_i \circ d_{r_i}^{-1} \circ a_i(p_{i-1}) \quad \text{for } i = 1, \ldots, n.$$

Write $f_i = d_{p_i}^{-1} \circ k_i \circ d_{r_i}^{-1} \circ a_i$, which, if assumption (A6) holds, is well defined.

We compose the f_i's to obtain a fixed-point problem:

$$p_i = f_i \circ f_{i-1} \circ \cdots \circ f_1 \circ f_n \circ \cdots \circ f_{i+1}(p_i), \quad \text{for } i = 1, \ldots, n. \tag{2.13}$$

Since the f_i's are monotonically decreasing by (A3) and (D3) and we are composing an odd number of functions, the composition in (2.13) is monotonically decreasing. It is also positive at 0 by (A3), (A4), (D2), and (D3). Therefore, for $i = 1, \ldots, n$, there is exactly one solution to Eqn. (2.13) in $\mathbb{R}^+$, so system (2.7) has a unique steady state in $\mathbb{R}^{2n}_+$. $\square$

We follow the notation in [20] and label this unique steady state as follows:

Definition 2.3.4. The **central steady state**, E_C, is the concentration vector

$$\left(d_{r_1}^{-1} \circ a_1(p_n^*), d_{r_2}^{-1} \circ a_2(p_1^*), \ldots, d_{r_n}^{-1} \circ a_n(p_{n-1}^*), p_1^*, \ldots, p_n^*\right), \tag{2.14}$$

where, for $i = 1, \ldots, n$, the quantity p_i^* solves Eqn. (2.13).

Remark 2.3.5. A solution to Eqn. (2.13) gives a steady state as in (2.14) regardless of whether n is even or odd because it solves a fixed-point problem derived from setting the equations of system (2.7) to zero.

2.3.1.2 Even-n Case

Below, we give various conditions under which the fixed-point problem in Eqn. (2.13) has a solution and consequently, guarantees when E_C is a steady state. First, however, we introduce another assumption on the degradation-rate functions.

(D4) $(d_{p_i})'(0) \neq 0$ and $(d_{r_i})'(0) \neq 0$ for all $i = 1, ..., n$.

Remark 2.3.6. Assumption (D4) is biologically reasonable as many commonly used degradation-rate functions satisfy (D4), e.g., linear degradation and Michaelis-Menten kinetics. However, there exist degradation processes that do not satisfy (D4). For example, consider a protein that is selected for degradation by dimerization with itself. If we model this scenario with a quadratic degradation term, then it will not satisfy assumption (D4).

With assumption (D4), we now prove various conditions under which system (2.7) admits a steady state.

Proposition 2.3.7. For system (2.7) with n even, if assumptions (A5), (A6), and (D4) hold, then E_C exists and is a steady state.

Proof. We follow the notation used in Proposition 2.3.3 and show that there exists a solution to the fixed-point problem from (2.13):

$$p_i = f_i \circ f_{i-1} \circ \cdots \circ f_1 \circ f_n \circ \cdots \circ f_{i+1}(p_i). \tag{2.15}$$

Note that all f_i's in Eqn. (2.15) are well defined by assumption (A6).

In Eqn. (2.15), we are composing an even number of strictly decreasing functions, so the composition is strictly increasing. We also know that the composition is positive at zero by (A2), (A3), (D2), and (D3). We will show that

$$\lim_{x\to\infty} (f_i \circ f_{i-1} \circ \cdots \circ f_1 \circ f_n \circ \cdots \circ f_{i+1})'(x) = 0.$$

This, along with the composition being positive at zero, will imply that E_C exists. We compute:

$$(f_i \circ f_{i-1} \circ \cdots \circ f_1 \circ f_n \circ \cdots \circ f_{i+1})'$$

$$= (f_i' \circ f_{i-1} \circ \cdots \circ f_1 \circ f_n \circ \cdots \circ f_{i+1})(f_{i-1}' \circ f_{i-2} \cdots \circ f_1 \circ f_n \circ \cdots \circ f_{i+1})$$
$$(f_{i-2}' \circ f_{i-3} \cdots \circ f_1 \circ f_n \circ \cdots \circ f_{i+1}) \cdots f_{i+1}'.$$

First, we show that $\lim_{x\to\infty} f_{i+1}'(x) = 0$. The following calculations are straightforward and follow from (A5), (D2), and (D4):

$$f'_{i+1}(x) = ((d_{p_{i+1}}^{-1})' \circ k_{i+1} \circ d_{r_{i+1}}^{-1} \circ a_{i+1}(x)) \cdot \\ (k'_{i+1} \circ d_{r_{i+1}}^{-1} \circ a_{i+1}(x)) \cdot ((d_{r_{i+1}}^{-1})' \circ a_{i+1}(x)) \cdot a'_{i+1}(x), \tag{2.16}$$

$$\lim_{x\to\infty} (d_{r_{i+1}}^{-1})' \circ a_{i+1}(x) = \lim_{x\to 0} (d_{r_{i+1}}^{-1})'(x) \\ = \frac{1}{(d_{r_{i+1}})'(0)} < \infty, \tag{2.17}$$

$$\lim_{x\to\infty} (k'_{i+1} \circ d_{r_{i+1}}^{-1} \circ a_{i+1}(x)) = k'_{i+1}(0) < \infty, \tag{2.18}$$

$$\lim_{x\to\infty} (d_{p_{i+1}}^{-1})' \circ k_{i+1} \circ d_{r_{i+1}}^{-1} \circ a_{i+1}(x) = \lim_{x\to 0} (d_{p_{i+1}}^{-1})'(x) \\ = \frac{1}{(d_{p_{i+1}})'(0)} < \infty. \tag{2.19}$$

It is easy to check that Eqns. (2.16)-(2.19) imply:

$$\lim_{x\to\infty} f'_{i+1}(x) = 0.$$

Now we show that for $k = i, ..., 1, n, ..., i+2$:

$$\lim_{x\to\infty} (f'_k \circ f_{k-1} \circ \cdots \circ f_{i+1})(x) < \infty.$$

Recall that $f_i = d_{p_i}^{-1} \circ k_i \circ d_{r_i}^{-1} \circ a_i$. Then, by (A5),

$$\lim_{x\to\infty} (f'_k \circ f_{k-1} \cdots \circ f_{i+1})(x) = (f'_k \circ f_{k-1} \cdots \circ d_{p_{i+1}}^{-1} \circ k_{i+1} \circ d_{r_{i+1}}^{-1})(0) < \infty.$$

Therefore,

$$\begin{aligned}
&\lim_{x\to\infty}(f_i \circ f_{i-1} \circ \cdots \circ f_1 \circ f_n \circ \cdots \circ f_{i+1})'(x) \\
&= \lim_{x\to\infty}(f_i' \circ f_{i-1} \circ \cdots \circ f_1 \circ f_n \circ \cdots \circ f_{i+1}) \cdot \\
&\quad \lim_{x\to\infty}(f_{i-1}' \circ f_{i-2} \cdots \circ f_1 \circ f_n \circ \cdots \circ f_{i+1}) \cdots \lim_{x\to\infty} f_{i+1}'(x) = 0.
\end{aligned}$$

Since i was arbitrary, each p_i has a solution, and E_C exists and by Remark 2.3.5 is a steady state. □

Proposition 2.3.8. Consider system (2.7) with n even. If $\alpha_i > 0$ for all $i = 1, ..., n$ and (A6) holds, then E_C exists and is a steady state.

Proof. The proof is similar to the proof of Proposition 2.3.7. Assumption (A6) implies that the inverses of $d_{r_i}(r_i)$ and $d_{p_i}(p_i)$ exist at $a_i(0)$ for all i. Also, by assuming that $\alpha_i > 0$, both

$$\lim_{x\to\infty}(d_{r_i}^{-1})' \circ a_i(x) \quad \text{and} \quad \lim_{x\to\infty}(d_{p_i}^{-1})' \circ k_i \circ d_{r_i}^{-1} \circ a_i(x)$$

are finite because $d_{p_i}'(\alpha_i), d_{r_i}'(\alpha_i) > 0$ by assumption (D3). □

We present a final sufficient condition for when E_C is a steady state in the even-n case. The condition is motivated by the following example.

Example 2.3.9. Consider the following generalized 2-gene repressilator model:

$$\begin{aligned}
\dot{r}_1 &= \frac{1}{1+p_2^2} - r_1^2 \\
\dot{r}_2 &= \frac{1}{1+p_1^2} - r_2^2 \\
\dot{p}_1 &= r_1 - p_1^2 \\
\dot{p}_2 &= r_2 - p_2^2.
\end{aligned}$$

This model fails the assumptions of Proposition 2.3.7, namely (D4), because the derivatives of the degradation-rate functions $d_{p_i} = p_i^2$ at zero are zero, and it fails those of Proposition 2.3.8

because $\alpha_1 = \alpha_2 = 0$. Nevertheless, E_C exists and is a steady state, because E_C is the solution to the following system:

$$p_1^4 = \frac{1}{1+p_2^2}$$
$$p_2^4 = \frac{1}{1+p_1^2}.$$

Finding the fixed point is equivalent to solving:

$$p^4 = \frac{1}{1+p^2}. \tag{2.20}$$

The left-hand side of Eqn. (2.20) is zero at zero and increases to ∞ while the right-hand side is greater than zero at zero and decreasing, so E_C exists. This phenomenon leads to our final result about E_C in the even-n case.

Proposition 2.3.10. Consider system (2.7) with n even or odd. Assume that all the degradation-rate functions d_{r_i} are equal ($= d_r$), all the degradation-rate functions d_{p_i} are equal ($= d_p$), all the transcription-rate functions a_i are equal ($= a$), and all the translational-rate functions k_i are equal ($= k$). If $\lim_{x\to\infty} k(x) > \delta^P$, where $\delta^P := \lim_{x\to\infty} d_p(x)$, then E_C exists and is a steady state.

Proof. Under the assumptions of the proposition, it is easy to check that computing E_C reduces to solving

$$a(p) = d_r \circ k^{-1} \circ d_p(p) \tag{2.21}$$

for $p \in \mathbb{R}^+$. The composition $d_r \circ k^{-1} \circ d_p(p)$ is well defined for all $p > 0$ by the assumption that $\lim_{x\to\infty} k(x) > \delta^P$. Also, the function $a(p)$ is decreasing, while the composition $d_r \circ k^{-1} \circ d_p(p)$ is increasing. Finally, $a(0) > d_r \circ k^{-1} \circ d_p(0) = 0$ by assumptions (A4) and (D2). Therefore, there is a solution $p \in \mathbb{R}^+$ to Eqn. (2.21), so E_C exists. □

Remark 2.3.11. The combinations of assumptions in Propositions 2.3.3-2.3.10 used to prove existence of E_C provide insight into possible repressilator design circuits. For example, a design circuit with a low-copy plasmid and a protein that is signaled for degradation through dimerization with itself could be problematic because the system may not have a steady state. Likewise, assumption

(A6)–used in the proofs of Propositions 2.3.3-2.3.8–requires that the maximal mRNA degradation rate "overcome" the maximal transcription rate. We revisit the theme of comparing degradation rates and synthesis rates when we address the stability of steady states in the next section.

2.3.2 Stability Analysis

For their model, Müller *et al.* proved general results about the stability of the central steady state by harnessing the fact that the matrix $J - \lambda I$, where J is the Jacobian of system (2.7) at E_C, is a circulant matrix. This matrix representation allowed the eigenvalues to be represented in terms of roots of unity, which in turn allowed for identifying general inequalities in the parameters that characterize stability. For our generalized repressilator model, however, the matrix $J - \lambda I$ does not reduce to a circulant matrix. Thus, we use different methods to characterize stability.

We begin with a few definitions.

Definition 2.3.12. Consider the generalized repressilator system (2.7). Let $x^* \in \mathbb{R}^{2n}_+$.

1. The **i-th mRNA degradation rate** at x^* is

$$\partial_i^R = \frac{\mathrm{d}d_{r_i}(r_i)}{\mathrm{d}r_i}\bigg|_{x^*}.$$

2. The **i-th protein degradation rate** at x^* is

$$\partial_i^P = \frac{\mathrm{d}d_{p_i}(p_i)}{\mathrm{d}p_i}\bigg|_{x^*}.$$

3. The **i-th degradation product** at x^* is

$$\mathcal{D}_i := \partial_i^R \partial_i^P.$$

4. The **total degradation product** at x^* is

$$\mathcal{D} := \prod_{i=1}^{n} \mathcal{D}_i,$$

where $\mathcal{D}_i$ is the i-th degradation product at x^*.

5. The **i-th synthesis product** at x^* is

$$\mathcal{K}_i := \left(\frac{\mathrm{d}k_i(r_i)}{\mathrm{d}r_i} \Big|_{x^*} \right) \left(\frac{\mathrm{d}a_i(p_{i-1})}{\mathrm{d}p_{i-1}} \Big|_{x^*} \right).$$

6. The **total synthesis product** at x^* is

$$\mathcal{K} := \prod_{i=1}^{n} \mathcal{K}_i,$$

where $\mathcal{K}_i$ is the i-th synthesis product at x^*.

When n is even, the total synthesis product is positive, because the even number of repression elements in the cycle results in a positive feedback system (see Section 2.2.4). In the odd-n case, the total synthesis product is negative, because the system is a negative feedback system (see Section 2.2.4). These differences play an important role in determining the stability of E_C.

2.3.2.1 Even-n Case

For system (2.1) with n even, Müller *et al.* found a condition on the derivative of the transcription-rate function that characterizes when the central steady state is stable. Here, we generalize that criterion to system (2.7) using $\mathcal{D}$ and $\mathcal{K}$.

Theorem 2.3.13. Consider system (2.7) with n even. A steady state x^* is locally asymptotically stable if and only if

$$\mathcal{D} > \mathcal{K}, \tag{2.22}$$

where $\mathcal{D}$ and $\mathcal{K}$ are evaluated at x^*.

Proof. It is easily checked that the characteristic polynomial of the Jacobian matrix of system (2.7) at x^* is

$$p(\lambda) = \prod_{j=1}^{n} (\lambda + \partial_j^R)(\lambda + \partial_j^P) - \mathcal{K}.$$

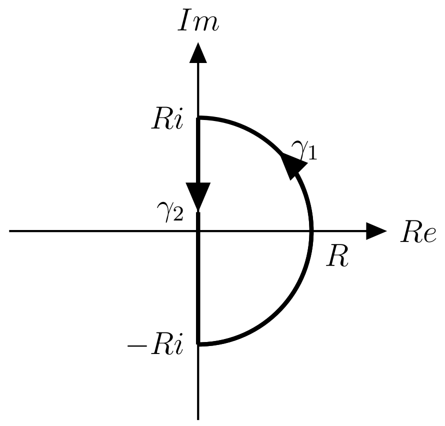


Figure 2.3: Contour Γ in proof of Theorem 2.3.13.

It follows that the constant term of p is $\mathcal{D} - \mathcal{K}$.

($\Longrightarrow$) We use the Routh-Hurwitz criterion. Assume that system (2.7) is stable at x^*. Then $\det(H_{n-1}) > 0$ and $\det(H_n) > 0$. However, $\det(H_n) = \det(H_{n-1}) \cdot (\mathcal{D} - \mathcal{K})$ implying that $\mathcal{D} - \mathcal{K} > 0$, i.e., $\mathcal{D} > \mathcal{K}$.

($\Longleftarrow$) We use Rouché's Theorem [46]. Write $p_1(z) = \prod_{j=1}^{n}(z + \partial_j^R)(z + \partial_j^P)$ and $p_2(z) = \mathcal{K}$. We will show that the number of zeros of $p(\lambda)$ in the right-hand half plane is equal to the number of zeros of p_1 in the right-hand half plane. Since all ∂_j's are positive by assumption (D3), there are no zeros of $p_1(z)$ in the right-hand half plane, so there are no zeros of $p(\lambda)$.

Consider the contour described by the semicircle of radius R, where R is sufficiently large, in the right-hand half plane along with the line segment connecting $-Ri$ and Ri on the imaginary axis. Call the contour Γ (Figure 2.3). We separate Γ into the semicircle, γ_1, and the line, γ_2. This is a closed contour in the complex plane. First, we show that $|p_1(z)| > |p_2(z)|$ on γ_1. We write $z = Re^{i\theta}$ on γ_1. Then

$$|p_1(z)| = |p_1(Re^{i\theta})| = \prod_{j=1}^{n} |Re^{i\theta} + \partial_j^R||Re^{i\theta} + \partial_j^P| \geq \prod_{j=1}^{n} |R - \partial_j^R||R - \partial_j^P|,$$

by the reverse triangle inequality. Call d the maximum of the degradation constants. Then

$$\prod_{j=1}^{n} |R - \partial_j^R||R - \partial_j^P| \geq \prod_{j=1}^{2n} |R - d|,$$

for sufficiently large R. Let $R' = 2d + 1$. Then for all $R \geq R'$,

$$\prod_{j=1}^{2n} |R - d| > d^{2n} \geq \mathcal{D}.$$

Therefore, for contours Γ with a sufficiently large radius, by assumption (2.22), the following inequalities hold on γ_1:

$$|p_1(z)| > \mathcal{D} > |\mathcal{K}| = |p_2(z)|.$$

Now all that is left to show is that $|p_1(z)| > |p_2(z)|$ on γ_2. On γ_2, we write $z = iy$ for $-R < y < R$. Then

$$|p_1(z)| = \prod_{j=1}^{n} |iy + \partial_j^R||iy + \partial_j^P| \geq \prod_{j=1}^{n} |Re(iy + \partial_j^R)||Re(iy + \partial_j^P)| = \mathcal{D}.$$

Therefore, again by assumption (2.22), the following holds on γ_2:

$$|p_1(z)| \geq \mathcal{D} > \mathcal{K} = |p_2(z)|.$$

The number of zeros of $p_1(z) - p_2(z)$ inside Γ is the same as the number of zeros of $p_1(z)$ inside Γ for all $R \geq R'$. Since $\partial_i^R, \partial_i^P > 0$ for all i, we know that there are no zeros of $p(\lambda)$ inside Γ for all $R \geq R'$. Therefore, there are no eigenvalues of the Jacobian with positive or zero real part, so the system is stable. $\square$

Theorem 2.3.13 has the following biological interpretation. Inequality (2.22) says that, in the long term, degradation is a more powerful process than synthesis. Thus, system (2.7) converges locally if and only if degradation is stronger than the combined synthesis of mRNA and protein.

2.3.2.2 Odd-n Case

Recall that, in Proposition 2.3.3, we proved when E_C exists and showed it is unique when n is odd. Below, we prove results towards finding a necessary and sufficient condition for stability of E_C in the odd-n case, like we have in the even case from Theorem 2.3.13. Our proofs use Hurwitz matrices because the inherent structure of the system when n is odd allows us to simplify the Routh-Hurwitz criterion. The main result of this section, Theorem 2.3.15, reduces the number of determinants of the Routh-Hurwitz criterion to check from $2n$ to $n-2$. We show at the end of the section, through an example system (2.1), that we cannot extend this result.

First, we discuss why the proof of Theorem 2.3.13 does not generalize to the odd-n case. Recall that, in this case, system (2.7) is a negative feedback loop and $\mathcal{K} < 0$, while in the even case, $\mathcal{K} > 0$. Thus, in the odd case, $\mathcal{D} > \mathcal{K}$ always holds, regardless of whether E_C is stable. Also, although $\mathcal{D} > 0 > \mathcal{K}$, we are not guaranteed that

$$\mathcal{D} > |\mathcal{K}|, \tag{2.23}$$

which is what we used in the proof of Theorem 2.3.13. If inequality (2.23) does hold, however, we conclude that the system is stable at E_C.

Theorem 2.3.14. Consider system (2.7) with n odd. If inequality (2.23) holds, then E_C is locally asymptotically stable.

Proof. The proof is the same as in the backwards direction of Theorem 2.3.13. □

We continue to solve the question of stability at E_C by using the structure of the system to reduce the number of Hurwitz matrices needed in the Routh-Hurwitz criterion. The idea is that the characteristic polynomial of the system is close to a polynomial known to have all negative real roots and so we will need to check fewer Hurwitz determinants.

Theorem 2.3.15. Consider system (2.7) with n odd, and let D_i denote the determinant of the i-th Hurwitz matrix of the Jacobian at E_C. Then E_C is locally stable if and only if $D_i > 0$ for all

$i = n+2, \ldots, 2n-1$.

Proof. We first show that, when n is odd, the first $n+1$ Hurwitz matrices calculated from the characteristic polynomial of the Jacobian at E_C always have positive determinant.

Recall from the proof of Theorem 2.3.13 that the characteristic polynomial of the Jacobian matrix at E_C is $p(\lambda) = \prod_{i=1}^{n}(\lambda + \partial_i^R)(\lambda + \partial_i^P) - \mathcal{K}$, where $\mathcal{K}$ is the total synthesis product from Definition 2.3.12. Since n is odd and so $\mathcal{K} < 0$, we rewrite this as $p(\lambda) = \prod_{i=1}^{n}(\lambda + \partial_i^R)(\lambda + \partial_i^P) + |\mathcal{K}|$. We introduce a new polynomial $q(\lambda) = \prod_{i=1}^{n}(\lambda + \partial_i^R)(\lambda + \partial_i^P)$.

In what follows, any quantity with a superscript p is constructed using $p(\lambda)$, and similarly for $q(\lambda)$. Notice that $p(\lambda)$ and $q(\lambda)$ both have degree $2n$, so there are $2n$ Hurwitz matrices H_i^p for $p(\lambda)$ and H_i^q for $q(\lambda)$. Also, all coefficients of $p(\lambda)$ and $q(\lambda)$ match except for the constant term. Therefore, every Hurwitz matrix constructed using only coefficients of $p(\lambda)$ that are not the constant term is equivalent to the corresponding Hurwitz matrix of $q(\lambda)$. We will use this fact below.

We now split the proof into two cases.

1. Case 1: $i = 1, ..., n$.

   From Definition 2.2.6, the coefficients of the polynomial that appear in H_i are indexed by $1, ..., 2i-1$. Therefore, $H_i^p = H_i^q$ for $i = 1, ..., n$, so $D_p^i > 0$ for $i = 1, ..., n$ because all roots of $q(\lambda)$ have negative real part.

2. Case 2: $i = n+1$.

   For this case, we examine the effect of the constant term of p on the determinant of H_{n+1}^p. Recall that $a_{2n}^p = a_{2n}^q + |\mathcal{K}|$ where a_{2n}^p and a_{2n}^q are the constant terms of p and q, respectively.

   Below, we use $A^{[a,b]}$ to denote the matrix A without row-a and column-b. The $(n+1)$st Hurwitz matrix of p is the following $(n+1) \times (n+1)$ matrix:

$$H^p_{n+1} = \begin{bmatrix} a_1 & a_0 & 0 & 0 & \dots & 0 \\ a_3 & a_2 & a_1 & a_0 & \dots & 0 \\ a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & a^p_{2n} & a_{2n-1} & \dots & a_{n+2} & a_{n+1} \end{bmatrix},$$

and H^q_{n+1} matches H^p_{n+1} at all entries except for entry $(n+1, 2)$, where it is the constant term a^q_{2n} rather than that of p. We compute $D^p_{n+1} = \det(H^p_{n+1})$ and $D^q_{n+1} = \det(H^q_{n+1})$ by expanding along the last row:

$$\begin{aligned} D^p_{n+1} = a^p_{2n} \det(H^{p,[n+1,2]}_{n+1}) - a_{2n-1} \det(H^{p,[n+1,3]}_{n+1}) + \dots \\ + a_{n+1} \det(H^{p,[n+1,n+1]}_{n+1}), \end{aligned} \tag{2.24}$$

and

$$\begin{aligned} D^q_{n+1} = a^q_{2n} \det(H^{q,[n+1,2]}_{n+1}) - a_{2n-1} \det(H^{q,[n+1,3]}_{n+1}) + \dots \\ + a_{n+1} \det(H^{q,[n+1,n+1]}_{n+1}). \end{aligned} \tag{2.25}$$

As the constant term is present only in the last row of H_{n+1}, the submatrices of H^p_{n+1} and H^q_{n+1} that exclude that row are equal. Combining this fact with Eqns. (2.24) and (2.25) gives

$$\begin{aligned} D^p_{n+1} - D^q_{n+1} &= a^p_{2n} \det(H^{p,[n+1,2]}_{n+1}) - a^q_{2n} \det(H^{q,[n+1,2]}_{n+1}) \\ &= (a^p_{2n} - a^q_{2n}) \det(H^{p,[n+1,2]}_{n+1}) = |\mathcal{K}| \det(H^{p,[n+1,2]}_{n+1}). \end{aligned} \tag{2.26}$$

To compute the determinant of the following matrix:

$$H_{n+1}^{p,[n+1,2]} = \begin{bmatrix} a_1 & 0 & 0 & \dots & 0 \\ a_3 & a_1 & a_0 & \dots & 0 \\ a_5 & a_3 & a_2 & a_1 & a_0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{2n-1} & a_{2n-3} & a_{2n-4} & \dots & a_{n-1} \end{bmatrix}$$

we expand about the first row, so $\det(H_{n+1}^{p,[n+1,2]}) = a_1 \det(A)$, where

$$A = \begin{bmatrix} a_1 & a_0 & 0 & 0 & 0 & 0 & \dots & 0 \\ a_3 & a_2 & a_1 & a_0 & 0 & 0 & \dots & 0 \\ a_5 & a_4 & a_3 & a_2 & a_1 & a_0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{2n-3} & a_{2n-4} & a_{2n-5} & a_{2n-6} & \dots & a_{n+1} & a_n & a_{n-1} \end{bmatrix}.$$

Notice that $A = H_{n-1}^p = H_{n-1}^q$, which has positive determinant by Case 1. Therefore, because $a_1 > 0$, $\det(H_{n+1}^{p,[n+1,2]}) = a_1 \det(A) > 0$. Since all roots of $q(\lambda)$ have negative real part, $D_{n+1}^q > 0$ by Theorem 2.2.7, so Eqn. (2.26) gives

$$D_{n+1}^p - D_{n+1}^q = |\mathcal{K}| \det(H_{n+1}^{p,[n+1,2]}) > 0 \implies D_{n+1}^p > D_{n+1}^q > 0.$$

Therefore, $D_{n+1}^p > 0$ and so the first $n+1$ determinants of the Hurwitz matrices constructed from $p(\lambda)$ are positive.

Since $D_{2n} = (\mathcal{D} - \mathcal{K})D_{2n-1}$ and $\mathcal{D} - \mathcal{K} > 0$ (as explained above Proposition 2.3.14), we conclude from Theorem 2.2.7 that E_C is locally stable if and only if $D_i > 0$ for all $i = n+2, \dots, 2n-1$. □

Corollary 2.3.16. For $n = 3$, system (2.7) is stable at E_C if and only if $D_5 > 0$.

Proof. Follows immediately from Theorem 2.3.15. □

Next, we recall the stability condition for E_C due to Müller *et al.* and compare it to the one in Theorem 2.3.15. Müller *et al.*'s criterion [20] is:

$$\frac{\beta}{(1+\beta)^2} < \frac{1 - S_c \cos(\pi/n)}{S_c^2 \sin^2(\pi/n)}, \tag{2.27}$$

where

$$S_c = -\alpha f'(E_C). \tag{2.28}$$

In system (2.7), it is easy to see that S_c equals $-\mathcal{K}_i$ at E_C. Therefore, we rewrite Eqn. (2.27) as:

$$\frac{\beta}{(1+\beta)^2} < \frac{1 + \mathcal{K}_i \cos(\pi/n)}{\mathcal{K}_i^2 \sin^2(\pi/n)}. \tag{2.29}$$

For $n = 3$, it is straightforward to check that inequality (2.29) is equivalent to:

$$(4 + 2\mathcal{K}_i)(1+\beta)^2 - 3\beta\mathcal{K}_i^2 > 0. \tag{2.30}$$

For system (2.1) with $n = 3$, by Corollary 2.3.16, the condition $D_5 > 0$ characterizes the same stability region in parameter space as inequality (2.30). This is surprising because D_5 under system (2.1) and $n = 3$ is a more complicated expression than the left-hand side in (2.30):

$$\begin{aligned} D_5 = \beta^2(&8\beta^{10}\mathcal{K}_i^3 + 64\beta^{10} + 144\beta^9\mathcal{K}_i^3 + 576\beta^9 + 792\beta^8\mathcal{K}_i^3 + 2304\beta^8 - 27\beta^7\mathcal{K}_i^6 \\ &+ 2184\beta^7\mathcal{K}_i^3 + 5376\beta^7 - 81\beta^6\mathcal{K}_i^6 + 3528\beta^6\mathcal{K}_i^3 + 8064\beta^6 - 81\beta^5\mathcal{K}_i^6 \\ &+ 3528\beta^5\mathcal{K}_i^3 + 8064\beta^5 - 27\beta^4\mathcal{K}_i^6 + 2184\beta^4\mathcal{K}_i^3 + 5376\beta^4 + 792\beta^3\mathcal{K}_i^3 \\ &+ 2304\beta^3 + 144\beta^2\mathcal{K}_i^3 + 576\beta^2 + 8\beta\mathcal{K}_i^3 + 64\beta). \end{aligned} \tag{2.31}$$

Next, we prove directly that these two inequalities define the same stability region when $\beta \in \mathbb{R}_{>0}$ and $\mathcal{K}_i \in \mathbb{R}$. Note that, by definition, $\mathcal{K}_i$ is always negative, but we show that even for $\mathcal{K}_i \in \mathbb{R}$ the two inequalities are equivalent.

Theorem 2.3.17 (Equivalence of the $n = 3$ stability conditions)**.** For $n = 3$ of system (2.1), inequality (2.30) holds for $\beta \in \mathbb{R}_{>0}$ and $\mathcal{K}_i \in \mathbb{R}$ if and only if $D_5 > 0$, where D_5 is the determinant of the Hurwitz matrix H_5 of the characteristic polynomial of the Jacobian matrix of (2.1) evaluated at E_C.

Proof. Let $f(\beta, \mathcal{K}_i) = (4 + 2\mathcal{K}_i)(1 + \beta)^2 - 3\beta\mathcal{K}_i^2$ denote the polynomial on the left-hand side of (2.30). We rename D_5, as in (2.31), the polynomial $g(\beta, \mathcal{K}_i)$. We must show that $f(\beta, \mathcal{K}_i)$ and $g(\beta, \mathcal{K}_i)$ are the same sign for all $\beta \in \mathbb{R}_{>0}$ and $\mathcal{K}_i \in \mathbb{R}_{<0}$.

It is straightforward to check, e.g., using `Maple`, that $g(\beta, \mathcal{K}_i) = f(\beta, \mathcal{K}_i)h(\beta, \mathcal{K}_i)$, where

$$\begin{aligned}
h(\beta, \mathcal{K}_i) = {} & 9\mathcal{K}_i^4\beta^8 + 6\mathcal{K}_i^3\beta^9 + 4\mathcal{K}_i^2\beta^{10} + 27\mathcal{K}_i^4\beta^7 + 30\mathcal{K}_i^3\beta^8 + 40\mathcal{K}_i^2\beta^9 \\
& - 8\mathcal{K}_i\beta^{10} + 27\mathcal{K}_i^4\beta^6 + 60\mathcal{K}_i^3\beta^7 + 144\mathcal{K}_i^2\beta^8 - 56\mathcal{K}_i\beta^9 + 16\beta^{10} \\
& + 9\mathcal{K}_i^4\beta^5 + 60\mathcal{K}_i^3\beta^6 + 260\mathcal{K}_i^2\beta^7 - 168\mathcal{K}_i\beta^8 + 112\beta^9 + 30\mathcal{K}_i^3\beta^5 \\
& + 260\mathcal{K}_i^2\beta^6 - 280\mathcal{K}_i\beta^7 + 336\beta^8 + 6\mathcal{K}_i^3\beta^4 + 144\mathcal{K}_i^2\beta^5 - 280\mathcal{K}_i\beta^6 \\
& + 560\beta^7 + 40\mathcal{K}_i^2\beta^4 - 168\mathcal{K}_i\beta^4 + 560\beta^6 + 4\mathcal{K}_i^2\beta^3 - 56\mathcal{K}_i\beta^4 + 336\beta^5 \\
& - 8\mathcal{K}_i\beta^3 + 112\beta^4 + 16\beta^3.
\end{aligned}$$

Because $g = fh$, any root of f is also a root of g. We will use this fact below.

Fix $\tilde{\beta} > 0$. Let $g_{\tilde{\beta}}(\mathcal{K}_i) := g(\tilde{\beta}, \mathcal{K}_i)$ and $f_{\tilde{\beta}}(\mathcal{K}_i) := f(\tilde{\beta}, \mathcal{K}_i)$. We rewrite $g_{\tilde{\beta}}$:

$$\begin{aligned}
g_{\tilde{\beta}}(\mathcal{K}_i) = {} & \mathcal{K}_i^6(-81\tilde{\beta}^6 - 81\tilde{\beta}^5 - 27\tilde{\beta}^4) + \mathcal{K}_i^3(8\tilde{\beta}^{10} + 144\tilde{\beta}^9 \\
& + 792\tilde{\beta}^8 + 2184\tilde{\beta}^7 + 3528\tilde{\beta}^6 + 3528\tilde{\beta}^5 + 2184\tilde{\beta}^4 \\
& + 792\tilde{\beta}^3 + 144\tilde{\beta}^2 + 8\tilde{\beta}) + C, \qquad (2.32)
\end{aligned}$$

where C is the sum of all the pure β terms in (2.31). It is easy to check that $C > 0$ when $\tilde{\beta} > 0$. Thus, we see from (2.32) that the polynomial $g_{\tilde{\beta}}$ has one sign change. Therefore, by Descartes' rule of signs, $g_{\tilde{\beta}}$ has at most one positive real root and at most one negative real root.

From (2.30), $f_{\tilde{\beta}}(\mathcal{K}_i)$ is a quadratic polynomial in $\mathcal{K}_i$ that is downward facing and has a positive

y-intercept namely, $(4(1+\tilde{\beta})^2)$. Therefore, $f_{\tilde{\beta}}$ has exactly two real roots, and thus, $g_{\tilde{\beta}}$ has exactly two real roots as well because $g = fh$ and, as noted above, $g_{\tilde{\beta}}$ has at most two real roots.

We label these two real roots r_1 and r_2 with $r_1 < r_2$. Since $g_{\tilde{\beta}}$ has even degree in $\mathcal{K}_i$ with a negative leading coefficient and a positive y-intercept, we know that $g_{\tilde{\beta}} > 0$ if and only if $\mathcal{K}_i$ is in the interval (r_1, r_2). It is straightforward to check that $f_{\tilde{\beta}}(\mathcal{K}_i)$ also is positive if and only if $\mathcal{K}_i$ is in the interval (r_1, r_2). Therefore, $f_{\tilde{\beta}} > 0$ if and only if $g_{\tilde{\beta}} > 0$. Our choice of $\tilde{\beta} > 0$ was arbitrary. Therefore, the two inequalities $D_5 > 0$ and (2.29) are equivalent. $\square$

Corollary 2.3.16 and the fact that Müller *et al.*'s criterion for system (2.1) is given by a single inequality led us to conjecture, in [21], that, when n is odd, the stability of E_C depends only on the penultimate Hurwitz determinant.

Conjecture 2.3.18. [21, Conjecture 1] For n odd, system (2.7) is stable at E_C if and only if $D_{2n-1} > 0$.

Evidence for Conjecture 2.3.18 is seen in the possible types of bifurcations of E_C in the odd case. We reorder the species as $r_1, p_1, r_2, p_2, ...$ to see that system (2.7) is a monotone system (see Section 2.2.4). In [38], Mallet-Paret and Smith showed that all omega-limit sets of monotone systems can be embedded in $\mathbb{R}^2$. Therefore, the possible bifurcations are stationary bifurcations or simple Hopf bifurcations. However, there cannot be stationary bifurcations because zero is never a root of the characteristic polynomial. Therefore, all bifurcations are simple Hopf bifurcations. Furthermore, from [35], at simple Hopf bifurcations, the following conditions hold: D_1, ..., $D_{2n-2} > 0$, and $D_{2n-1} = D_{2n} = 0$. This reasoning is not sufficient to prove the conjecture, however, because there could be a point in parameter space where E_C is unstable but nevertheless $D_{2n-1} > 0$. In fact, the conjecture turns out to be false as seen by the example presented in the next section.

2.3.3 Counterexample to Conjecture

Consider the following 5-gene repressilator system.

$$\begin{aligned}
\dot{r}_1 &= \frac{\alpha}{1+p_5^8} - r_1, \quad \dot{p}_1 = r_1 - p_1 \\
\dot{r}_2 &= \frac{\alpha}{1+p_1^8} - r_2, \quad \dot{p}_2 = r_2 - p_2 \\
\dot{r}_3 &= \frac{\alpha}{1+p_2^8} - r_3, \quad \dot{p}_3 = r_3 - p_3 \\
\dot{r}_4 &= \frac{\alpha}{1+p_3^8} - r_4, \quad \dot{p}_4 = r_4 - p_4 \\
\dot{r}_5 &= \frac{\alpha}{1+p_4^8} - r_5, \quad \dot{p}_5 = r_5 - p_5.
\end{aligned} \tag{2.33}$$

The structure of system (2.33) comes directly from system (EL) with $\beta = 1$ and $\alpha_0 = 0$. For now, we use α as a placeholder, but later, we will show that there exists an α such that $D_7 < 0$ and $D_9 > 0$, which will prove Conjecture 2.3.18 false.

First, we know that there is a unique, positive steady state of system (2.33) given by the vector of length 10 with all entries r^* where r^* is the solution to the fixed point problem

$$r = \frac{\alpha}{1+r^8}.$$

For all $i = 1, \ldots, 5$, the i-th synthesis rate of system (2.33) is

$$\mathcal{K}_i = \frac{-8\alpha(r^*)^8}{(1+(r^*)^8)^2}. \tag{2.34}$$

The total synthesis product, $\mathcal{K}$, is then $\mathcal{K}_i^5$.

With the notation in place, we give the Jacobian matrix, A, of system (2.33) at r^*.

$$A = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathcal{K}_i \\ 0 & -1 & 0 & 0 & 0 & \mathcal{K}_i & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & \mathcal{K}_i & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & \mathcal{K}_i & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & \mathcal{K}_i & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

The characteristic polynomial of A is

$$\begin{aligned} p(\lambda) &= (\lambda+1)^{10} - \mathcal{K} \\ &= \lambda^{10} + 10\lambda^9 + 45\lambda^8 + 120\lambda^7 + 210\lambda^6 + 252\lambda^5 + 210\lambda^4 + 120\lambda^3 + 45\lambda^2 + 10\lambda + 1 - \mathcal{K}. \end{aligned} \tag{2.35}$$

Using the coefficients in (2.35), we compute H_7 and H_9.

$$H_7 = \begin{bmatrix} 10 & 1 & 0 & 0 & 0 & 0 & 0 \\ 120 & 45 & 10 & 10 & 0 & 0 & 0 \\ 252 & 210 & 120 & 45 & 10 & 1 & 0 \\ 120 & 210 & 252 & 210 & 120 & 45 & 10 \\ 10 & 45 & 120 & 210 & 252 & 210 & 120 \\ 0 & 1-\mathcal{K} & 10 & 45 & 120 & 210 & 252 \\ 0 & 0 & 0 & 1-\mathcal{K} & 10 & 45 & 120 \end{bmatrix}$$

$$H_9 = \begin{bmatrix} 10 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 120 & 45 & 10 & 10 & 0 & 0 & 0 & 0 & 0 \\ 252 & 210 & 120 & 45 & 10 & 1 & 0 & 0 & 0 \\ 120 & 210 & 252 & 210 & 120 & 45 & 10 & 1-\mathcal{K} & 0 \\ 10 & 45 & 120 & 210 & 252 & 210 & 120 & 45 & 10 \\ 0 & 1-\mathcal{K} & 10 & 45 & 120 & 210 & 252 & 210 & 120 \\ 0 & 0 & 0 & 1-\mathcal{K} & 10 & 45 & 120 & 210 & 252 \\ 0 & 0 & 0 & 0 & 0 & 1-\mathcal{K} & 10 & 45 & 120 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\mathcal{K} & 10 \end{bmatrix}$$

If we substitute $\mathcal{K}_i = -3$ or $\mathcal{K} = -243$, then the determinant of H_7 is -3566730365440. If we substitute the same value for $\mathcal{K}$ into H_9, then the determinant of H_9 is 30502945385685152. Thus, we have found a total synthesis product that gives a counterexample to Conjecture 2.3.18. Now, it is left to show that there exists an $\alpha > 0$ such that $\mathcal{K}_i = -3$.

Recall from Eqn. (2.34) that

$$\mathcal{K}_i = \frac{-8\alpha(r^*)^8}{(1+(r^*)^8)^2}.$$

Therefore, to find a counterexample to Conjecture 2.3.18, we need an α and r^* such that

$$r^* = \frac{\alpha}{1+(r^*)^8} \quad \text{and} \quad 3 = \frac{8\alpha(r^*)^7}{(1+(r^*)^8)^2}$$

or

$$r^*(1+(r^*)^8) = \alpha \quad \text{and} \quad 3(1+(r^*)^8)^2 = 8\alpha(r^*)^7. \tag{2.36}$$

We use the symbolic toolbox in MATLAB to find positive real solutions to (2.36). We find values $r^* \approx 0.944693235528633$ and $\alpha \approx 1.491708014434$ satisfy (2.36). To show this result rigorously, we plot the curves of condition (2.36) and show that their respective graphs intersect in the positive quadrant (Figure 2.4).

Conjecture 2.3.18 worked towards a necessary and sufficient condition for stability of the

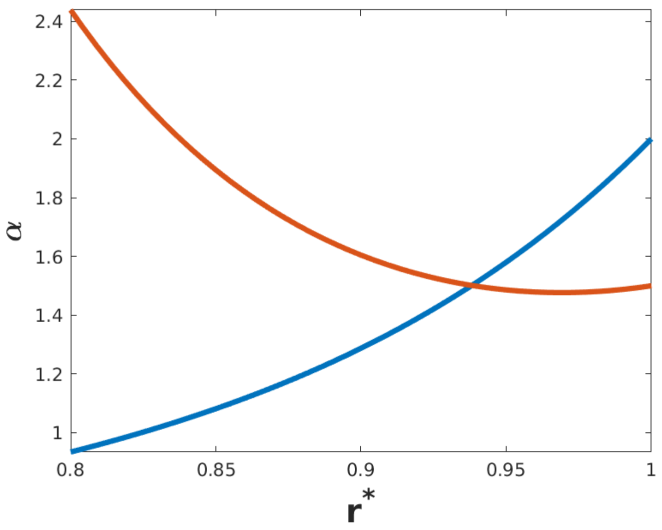


Figure 2.4: Plot of the curves given in condition (2.36). The blue curve plots points (r^*, α) that satisfy the fixed point problem given by the first equation in (2.36). The red curve plots points (r^*, α) that satisfy the second equation in (2.36) that guarantees that $\mathcal{K}_i = -3$.

unique steady state of the generalized repressilator with an odd number of genes. The counterexample given by system (2.33) shows that the condition $D_{2n-1} < 0$ is not necessary to conclude that E_C is unstable. Thus, the strongest result regarding the stability of E_C when the system has an odd number of genes continues to be Theorem 2.3.15.

Moreover, the counterexample is significant in that the system used for the counterexample is the previous model of Müller *et al*. Since system (2.33) follows directly from system (2.1), we cannot reduce the assumptions of Conjecture 2.3.18 to the previous system and then prove the conjecture true. I.e., the counterexample also disproves the following conjecture.

Conjecture 2.3.19. For $n > 3$ odd, system (2.1) is stable at E_C if and only if $D_{2n-1} > 0$.

When $n = 3$, the structure of system (2.1) is unique and allows for the sign of the second to last Hurwitz determinant to be determined completely by the simpler condition given by Müller *et al.* An interesting open question is to determine how inequality (2.27) by Müller *et al*. relates to $D_{n+2}, \ldots, D_{2n-1}$.

Finally, the counterexample is significant in that, for a 5-gene repressilator system, D_7 was the determinant that characterized stability. Since Theorem 2.3.15 already guarantees $D_1, \ldots D_{n+1} > 0$, we cannot extend Theorem 2.3.15 to include $D_{n+2} > 0$. Recall that Theorem 2.3.15, in the case of $n = 5$ reduced the Routh-Hurwitz criterion from checking all $D_i > 0$ for $i = 1, \ldots, 10$ to only checking $D_i > 0$ for $i = 7$, 8, and 9. Since it was D_7 that went negative while D_9 stayed positive, Theorem 2.3.15 cannot be extended in terms of the determinants to check from the Routh-Hurwitz criterion.

2.3.4 Asymptotic Behavior

Finally, we prove a result about the global dynamics of system (2.7), which is similar to Proposition 2.2.5, by using the result on monotone systems given in [38] recalled above in Proposition 2.2.9, Section 2.2.4.

Theorem 2.3.20. For n odd, system (2.7) has the following properties: (i) Every orbit converges to E_C or to a periodic orbit. (ii) If E_C is unstable, then there exists a periodic-orbit attractor.

Proof. It is straightforward to check that the proof is the same as that of Theorem 2 in [20], which uses the Main Theorem from [38]. We note that we can rule out the third option of the Main Theorem in [38] because E_C is unique, so there are no heteroclinic or homoclinic orbits. □

Theorem 2.3.20 is significant biologically because it shows the species concentrations of the repressilator constructed with an odd number of genes will either stabilize to the steady state value or to a limit-cycle.

2.4 Discussion

This work advances the theoretical study of cyclic gene repression by generalizing the current repressilator models. First, we permit more transcription-rate functions than the traditional single-step binding function. We require only that these functions satisfy a few properties that agree with current biological knowledge. We also broaden the possible degradation terms beyond first-order degradation. Again, we require only that these functions satisfy certain biological assumptions. Finally, we assume first-order translation rates but allow them to vary among mRNAs.

Our new system retains many advantageous qualitative properties of the previous repressilator after these generalizations. We proved, for instance, that the system with an odd number of genes has a unique steady state, called the central steady state. We also showed that the system with an odd number of genes converges to the central steady state or to a periodic orbit. We worked towards a necessary and sufficient condition for when the central steady state is stable.

For the even case, we characterized when the central steady state exists. We also give a biological criterion for when a steady state is stable. However, at the level of generality we propose, we cannot generalize the results of Müller *et al.* regarding the possible number of steady states. For specific choices of degradation-rate and transcription-rate functions, one can, however, analyze the limiting dynamics of system (2.7) with n even by using the Poincaré-Bendixson Theorem for monotone systems given in [38].

In summary, we now better understand stability and limiting dynamics of the repressilator system for a wide range of biologically relevant degradation-rate and transcription-rate functions. We hope that our results will encourage theoretical and experimental biologists to broaden the possible degradation-rate and transcription-rate functions used to model the repressilator and other gene regulatory networks. Finally, we expect that allowing general functions for these terms will generate more accurate and predictive models of not only the repressilator but genetic repression in general. To investigate how incorporating general functions will affect dynamics, in Section 3, we derive a new transcription-rate function and investigate the qualitative and quantitative differences between a model using the Hill function for transcription and a model using our new function for transcription.

3. TRANSCRIPTION-RATE FUNCTION FROM SUCCESSIVE BINDING[1]

3.1 Introduction

The discovery that circadian rhythms are conserved across many organisms–from cyanobacteria [47, 48] to the most complex mammals [49, 50]–led scientists to hypothesize that cells within organisms have self-sustaining molecular clocks that are then synchronized throughout the organism. Consequently, mathematicians used mathematical models to test whether hypothesized molecular mechanisms were capable of generating oscillations. For instance, in 1965, Goodwin presented the following mathematical model of a molecular feedback loop where a gene's product represses production of the gene itself ($X \to Y \dashv X$) [44]:

$$\begin{aligned} \frac{dX}{dt} &= \frac{a}{A + kY} - b, \\ \frac{dY}{dt} &= \alpha X - \beta. \end{aligned} \tag{3.1}$$

Here, the variable X is the concentration of the mRNA; the variable Y is the concentration of the protein. The other quantities are parameters. A major feature of this model is the appearance of Y in the denominator of the synthesis component of $\frac{dX}{dt}$ in (3.1). This denominator is due to the assumption that the way the repressor protein acts on DNA is similar to how inhibitors work on enzymes. Goodwin claimed that system (3.1) models a biological oscillator with amplitudes in Y much larger than those in X.

In 1968, Griffith generalized Goodwin's model by allowing for a general exponent of Y in the denominator of Eqn. (3.1) [51]. Griffith's model is

$$\begin{aligned} \frac{dX}{dt} &= \frac{a}{1 + KY^m} - bX, \\ \frac{dY}{dt} &= cX - dY. \end{aligned} \tag{3.2}$$

[1]Based on Sections 3 and 4 of "Revisiting a synthetic intracellular regulatory network that exhibits oscillations," by J. Tyler, A. Shiu, and J. Walton, 2019. *J MATH BIOL*, 78:2341-2368. Used with permission.

Here, the variables X and Y are the same as in system (3.1). The parameters a, b, c, and d are all positive constants. Recall from Remark 2.3.1 that the exponent m is called the *Hill coefficient*.

Griffith argued that, although computer simulations are helpful, mathematicians must approach these models with a more rigorous analysis. Thus, he used results of nonlinear dynamics to analyze model (3.2). Griffith's analysis showed that there is one limit cycle if and only if $m > 8$. These predictions were compared against computer simulations and held true. Notice that this result contradicts that of Goodwin who had found a limit cycle when $m = 1$. Apparently, Goodwin later considered his own result to have "arisen erroneously" [51].

Remark 3.1.1. The Hill coefficient introduced by Griffith has a clear biological interpretation. The exponent corresponds to the number of intermediate steps that are assumed to occur rapidly or, in the case of repression, the number of repressor proteins that rapidly bind to the promoter [10, 52]. In the case Goodwin considered, the Hill coefficient corresponds to the number of copies of a particular protein that are needed to bind to its own gene's promoter to inhibit transcription. Recall from above that Griffith showed, for one protein repressing its own gene's transcription, system (3.1) requires a Hill coefficient of eight to exhibit oscillations. Thus, eight repressors must bind to the promoter for the protein abundance to oscillate. However, a total of eight transcription factors is an unprecedented number in biology. In Section 4, we further investigate how oscillatory quantities (e.g., period, cost of protein production, skewness, etc.) of a model of the repressilator (Section 2) vary with respect to the Hill coefficient.

Since 1968, the Hill function derived by Griffith has been used extensively to model repression of gene transcription by a repressor and arises from the following "single-step assumptions" [45]:

1. On the promoter, either no repressor proteins are bound and transcription occurs, or repressors proteins are bound to all binding sites and no transcription occurs.

2. The repressor protein binds rapidly to the promoter.

These assumptions are restrictive because they require that all repressors bind to a gene's promoter instantaneously. Thus, the model does not incorporate information such as the time taken for the repressor proteins to bind.

Accordingly, we advocate for changing how we model repression. We introduce the following alternate set of assumptions, similar to those given in [45]:

1. There are m binding sites on a promoter, and the repressor proteins bind in order from sites 1 to m.
2. Transcription cannot occur if m repressor proteins are bound to the promoter. Transcription can occur in all other cases.
3. The repressor protein binds rapidly to the promoter.
4. Repressor proteins bind to the m binding sites at varying rates.

We label these assumptions the *successive-binding assumptions* and use them to derive a new transcription-rate function in Section 3.2. One major difference between the successive-binding and the single-step binding assumptions is that, in the successive-binding assumptions, the repressor proteins bind in succession; i.e., the repressor proteins do not "instantaneously" bind to the promoter together as in the single-step binding assumptions. Successive binding promotes qualitative and quantitative differences between models that incorporate the functions derived from the two lists of binding assumptions. In Section 3.3, we investigate these differences using the repressilator (Section 2) as a foundational model. Finally, in Section 3.4, we conjecture a theoretical result comparing the amplitudes of proteins of a repressilator model with the Hill function and a corresponding model with our new transcription-rate function.

3.2 New Transcription-Rate Function from Successive Binding

Below, we derive a new transcription-rate function from the successive-binding assumptions listed in Section 3.1. The derivation proceeds in two steps. First we derive the function that models binding of the proteins to the promoter (Section 3.2.1). Then, using this new successive-binding function, we derive the actual transcription-rate function (Section 3.2.2).

3.2.1 Successive-Binding Function

First, we recall the assumptions for successive binding introduced in Section 3.1.

1. There are m binding sites on a promoter, and the repressor proteins bind in order from sites 1 to m.
2. Transcription cannot occur if m repressor proteins are bound to the promoter. Transcription can occur in all other cases.
3. The repressor protein binds rapidly to the promoter.
4. Repressor proteins bind to the m binding sites at varying rates.

These assumptions are adapted from [45, Chapter 2] where Forger presents three models of repression. The model we are interested in is his Model "a": A Model for Transcription Regulation with Independent Binding Sites.

Here, we present the reaction mechanism. Let **G** denote the gene; and **P** the repressor protein acting on the gene. We write the i-th *gene-repressor complex*–the complex consisting of i repressor proteins, **P**, bound to the gene promoter **G**–as $\mathbf{C}^{(i)}$. The *successive-binding reaction mechanism* is

$$\begin{aligned} \mathbf{G} + \mathbf{P} &\rightleftharpoons \mathbf{C}^{(1)} \\ \mathbf{C}^{(1)} + \mathbf{P} &\rightleftharpoons \mathbf{C}^{(2)} \\ &\vdots \\ \mathbf{C}^{(m-1)} + \mathbf{P} &\rightleftharpoons \mathbf{C}^{(m)}. \end{aligned} \tag{3.3}$$

Assumption 1 presumes that the promoter has m binding sites and that repressors bind in order from site 1 to m, so the mechanism has m possible gene-repressor complexes $\mathbf{C}^{(1)}, ..., \mathbf{C}^{(m)}$. We will derive the binding functions $c^{(i)}$ for complex $\mathbf{C}^{(i)}$ that model the rate of repressor protein binding to the gene promoter as a function of the total gene concentration and concentration of the repressor protein present. We proceed with this derivation below.

Assumption 3 allows us to use the *quasi steady state assumption*–after an initial fast transient, some of the dependent variables can be regarded as in steady state [53]–on the concentrations of the gene-repressor complexes to derive the binding function. Hence, the binding function for $\mathbf{C}^{(1)}$ is

$$c^{(1)} = \frac{gp}{K_1}, \tag{3.4}$$

where g is the free gene concentration, p is the concentration of the repressor, and K_1 is a dissociation constant. In what follows, dissociation constants for each gene are distinct because of Assumption 4. We use the function (3.4) to write the binding function for $\mathbf{C}^{(2)}$:

$$c^{(2)} = \frac{pc^{(1)}}{K_2} = \frac{gp^2}{K_1K_2},$$

where K_2 is another dissociation constant. We continue this process to get a general formula for the binding function of the j-th complex:

$$c^{(j)} = \frac{gp^j}{K_1K_2\cdots K_j}, \tag{3.5}$$

where $K_1, \ldots, K_j$ are all dissociation constants.

Conservation of mass for genes is given by

$$\bar{g} = g + c^{(1)} + c^{(2)} + \cdots + c^{(m)} \tag{3.6}$$

where $\bar{g}$ is the total gene amount. This conservation equation differs from the conservation equation arising from single-step binding. Under single-step binding, the genes are either free or consumed in the final gene-repressor complex, leading to the conservation equation:

$$\bar{g} = g + c^{(m)}.$$

We desire a binding function that depends only on the protein product concentration and the total gene concentration. To obtain such a function, we must first solve for $c_i^{(m)}$ in terms of p using Eqns. (3.5) and (3.6).

$$c^{(m)} = \frac{(\bar{g} - c^{(1)} - \cdots - c^{(m-1)} - c^{(m)})p^m}{K_1 K_2 \cdots K_m}$$

$$\implies c^{(m)} = \frac{\bar{g}p^m}{K_1 K_2 \cdots K_m} - \frac{c^{(m)}p}{K_1} - \cdots - \frac{c^{(m)}p^{m-1}}{K_1 K_2 \cdots K_{m-1}} - \frac{c^{(m)}p^m}{K_1 K_2 \cdots K_m}$$

$$\implies c^{(m)} \left(1 + \frac{p}{K_1} + \frac{p^2}{K_1 K_2} + \cdots + \frac{p^m}{K_1 K_2 \cdots K_m}\right) = \frac{\bar{g}p^m}{K_1 K_2 \cdots K_m}$$

$$\implies c^{(m)} \left(\frac{K_1 K_2 \cdots K_m + K_2 \cdots K_m p + \cdots + K_{m-1}p^{m-1} + p^m}{K_1 K_2 \cdots K_m}\right)$$

$$= \frac{\bar{g}p^m}{K_1 K_2 \cdots K_m}$$

$$\implies c^{(m)} = \frac{\bar{g}p^m}{\sum_{j=0}^m ((\prod_{l>j} K_l)p^j)}.$$

Similarly, we obtain $c^{(j)}$:

$$c^{(j)} = \frac{(\prod_{\ell>j} K_\ell)\bar{g}p^j}{\sum_{j=0}^m ((\prod_{\ell>j}^m K_\ell)p^j)}. \tag{3.7}$$

We simplify notation by letting $B(p) = \sum_{j=0}^m ((\prod_{\ell>j} K_\ell)p^j)$ and $A^{(j)}(p) = (\prod_{\ell>j}^m K_\ell)p^j$, so that:

$$B(p) = \prod_{j=1}^m K_j + \sum_{j=1}^m A^{(j)}(p). \tag{3.8}$$

Therefore, we rewrite Eqn. (3.7), the *successive-binding function*, as

$$\boxed{c^{(j)} = \frac{\bar{g}A^{(j)}(p)}{B(p)} \text{ for } j = 1, \ldots, m.} \tag{3.9}$$

3.2.2 Transcription-Rate Function Obtained from Successive-Binding Function

We assume as in [20] that the transcription rate $a(p)$ depends linearly on the free gene concentration g given by the two cases

$$g = \bar{g} \implies a = \bar{g}, \tag{3.10}$$

and

$$g = 0 \implies a = \delta \bar{g}. \tag{3.11}$$

Here, following Müller *et al.* [20], the parameter δ denotes the ratio of repressed to unrepressed transcription. Case (3.10) assumes that, if the gene is free of any repressors, then transcriptional activity will occur proportional to the total gene concentration. Case (3.11) assumes that, if m repressors are bound to the gene, then transcriptional activity will occur proportional to the constant δ.

From cases (3.10) and (3.11), the transcription-rate a is given by

$$a = (1 - \delta) g + \delta \bar{g}.$$

We use Eqns. (3.6) and (3.9) to rewrite a:

$$a = \bar{g} \left[(1 - \delta) \left(1 - \frac{A^{(1)}(p) + A^{(2)}(p) + \cdots + A^{(m)}(p)}{B(p)} \right) + \delta \right]. \tag{3.12}$$

Using Eqn. (3.8), we rewrite Eqn. (3.12) as

$$a = \bar{g} \left[\frac{(1 - \delta) \prod_{j=1}^{m} K_j}{B(p)} + \delta \right].$$

To simplify notation, let us write

$$S(p) := \frac{\prod_{j=1}^{m} K_j}{B(p)}. \tag{3.13}$$

Then, from Eqns. (3.12) and (3.13), the derived transcription-rate function is:

$$\boxed{a_i = \bar{g}[(1 - \delta) S_i(p_{i-1}) + \delta].} \tag{3.14}$$

It is straightforward to check that Eqn. (3.14) satisfies assumptions (A1)-(A4), and hence is a valid transcription-rate function.

Theorem 3.2.1. The transcription-rate function arising from the successive-binding mechanism, given by Eqn. (3.14), satisfies assumptions (A1)-(A4) of the generalized repressilator model (Section 2).

Theorems 2.3.3 and 3.2.1 immediately yield the following corollary.

Corollary 3.2.2. Consider the repressilator system from Section 2 with n odd and transcription-rate functions $a_i(p_{i-1})$ in Eqn. (3.14), that is, arising from the successive-binding mechanism. Then the central steady state E_C exists and is the unique, positive steady state.

Remark 3.2.3. Forger, in [45], simplifies Eqn. (3.13) by assuming that the dissociation constant, K_j, is the same across each reaction in the successive-binding mechanism (3.3). Hence, his version of Eqn. (3.13) is:

$$S(p) = \frac{K^m}{(K+p)^m}.$$

3.3 Numerical Comparison of Models Arising from Hill Functions vs. Successive-Binding Transcription-Rate Functions

Below, we numerically compare a model using the traditional single-step binding assumption for transcription and another model constructed using the successive-binding assumption. Specifically, we show that the amplitudes and periods of the oscillations can differ widely (see Figures 3.1 and 3.2).

The first model is the following three-gene repressilator system:

$$\begin{aligned}
\dot{r}_1 &= \frac{k_1}{1+p_3^h} - r_1, & \dot{p}_1 &= 3(r_1 - p_1) \\
\dot{r}_2 &= \frac{k_2}{1+p_1^h} - r_2, & \dot{p}_2 &= 3(r_2 - p_2) \\
\dot{r}_3 &= \frac{k_3}{1+p_2^h} - r_3, & \dot{p}_3 &= 3(r_3 - p_3).
\end{aligned} \tag{SS}$$

Model (SS) (for single-step) is constructed using the single-step binding assumption for each transcription-rate function, and the Hill coefficients, h, are assumed to be equal.

In contrast, the second model considered is:

$$\begin{aligned}
\dot{r}_1 &= \frac{k_1}{(1+p_3)^h} - r_1, & \dot{p}_1 &= 3(r_1 - p_1) \\
\dot{r}_2 &= \frac{k_2}{(1+p_1)^h} - r_2, & \dot{p}_2 &= 3(r_2 - p_2) \\
\dot{r}_3 &= \frac{k_3}{(1+p_2)^h} - r_3, & \dot{p}_3 &= 3(r_3 - p_3).
\end{aligned} \tag{SB}$$

Model (SB) (for successive-binding) is constructed using the successive-binding assumption for each transcription-rate function (Eqn. (3.14)), and, like model (SS), the Hill coefficients, h, are assumed to be equal. Note from systems (SS) and (SB) that the two models are equivalent in the degradation and translation components.

Remark 3.3.1. In the subsequent analysis, we consider two scenarios: varying the Hill coefficient while fixing all other parameters as correct and varying the transcription rate while fixing all other parameters as correct. When varying the transcription rate, we fix the Hill coefficient as 3. Using Corollary 2.3.16, we compute the Hopf bifurcation of model (SS) as $k \approx 1.8466$ and the Hop bifurcation of model (SB) as $k \approx 6.2964$. When varying the Hill coefficient, we fix the transcription rate at 10. In this case, again using Corollary 2.3.16, we estimate the Hopf bifurcation of model (SS) as $h \approx 1.8043$ and the Hopf bifurcation of model (SB) as $h \approx 2.5717$. **The code we wrote to estimate the Hopf bifurcation values appears in Appendix A, Section A.1.1.**

3.3.1 Amplitude

Here, we compare the amplitudes of models (SS) and (SB). For the first numerical comparison, we vary the Hill coefficient, h, from 2 to 50 while keeping all other parameters fixed. For both models (SS) and (SB), we numerically solve the system until it reaches a steady state or a limit cycle. Then, we compute the amplitude of protein 1 by evaluating the difference of the maximum and minimum protein 1 abundance. Figure 3.1(a) shows the amplitudes of the first protein concentration with respect to the Hill coefficient (sampled at every two-tenth value–1, 1.2, 1.4, etc.) for models (SS) (black) and (SB) (red). All computations were performed in `MATLAB`.

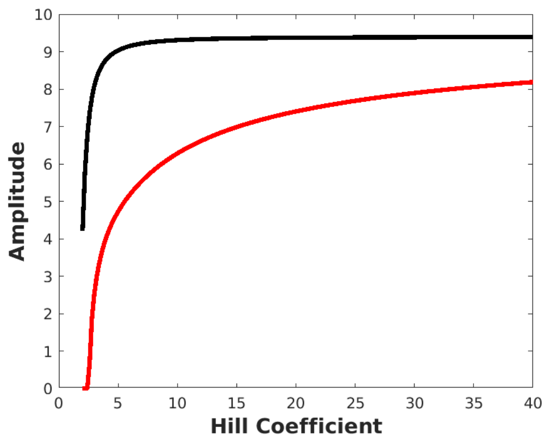
As shown in Figure 3.1(a), the amplitude of model (SS) increases to an order of magnitude larger than the amplitude of model (SB). This, along with a smaller Hopf bifurcation of model (SS) than model (SB) with respect to the Hill coefficient (see Remark 3.3.1), suggests that the traditional transcription-rate function allows for oscillations to occur at smaller Hill coefficients than for our newly derived transcription-rate function. This means, in terms of the biology, that under the single-step binding assumption, oscillations can occur when there are fewer repressors binding to the gene promoter. However, incorporating intermediate steps into the repressor-promoter interactions (like in the successive-binding assumption) leads to more repressors required to produce oscillations.

Next, we conducted a numerical comparison that fixed all parameters ($h = 3$) while letting the transcription rates, k_1, k_2, and k_3, vary. In order to plot the amplitudes, we assume that $k_1 = k_2 = k_3 = k$ and let k vary from 3 to 50. Again, we sample k at every two-tenth interval and numerically solve both models to convergence to the steady state or the limit cycle. We then compute the amplitudes as in the first comparison. Figure 3.1(b) shows the amplitudes of the first protein concentration with respect to the transcription rate for both models. Similar to the first comparison, model (SS) amplitudes are significantly different from those of model (SB), and in fact, reach an order of magnitude difference (Figure 3.1(b)).

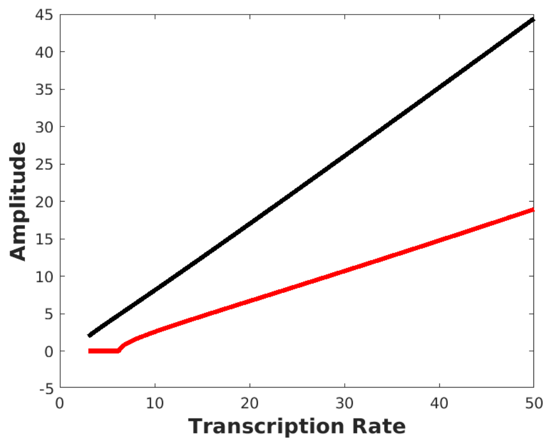
These and other numerical simulations support the claim that the amplitude of a model constructed using the successive-binding assumption will be smaller than the amplitude of a model constructed using the single-step binding assumption, all other components being equal. As amplitudes are an important quantity of oscillations of a system, care should therefore be taken when considering appropriate models of gene repression and transcription or when fitting models to actual data.

3.3.2 Period

Similar to the amplitude, the two transcription-rate functions yield dramatically different periods. To compare, we compute the periods of models (SS) and (SB), again using `MATLAB`. First, we fix all parameters except the Hill coefficient, h. Again, we let h vary from 2 to 50 and sample h

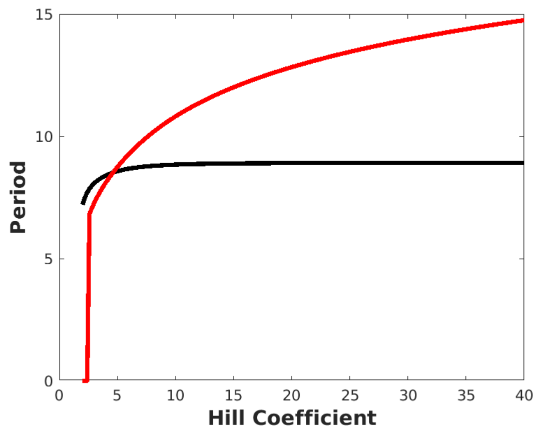


(a)

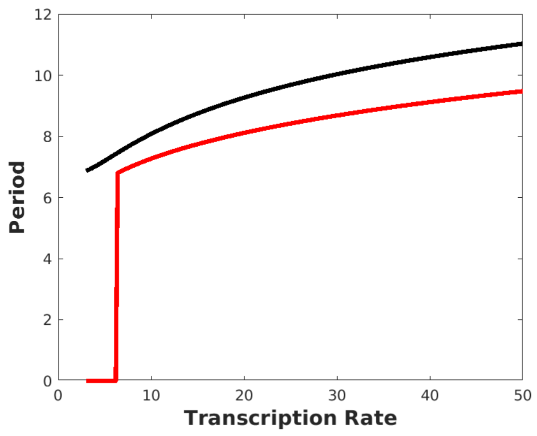


(b)

Figure 3.1: (a) Protein 1 concentration amplitudes for models (SS) (blue curve) and (SB) (red curve) with respect to the Hill coefficient. We fixed the transcription rates at the same value, $k = 10$. (b) Amplitudes of the concentration of protein 1 for models (SS) (blue) and (SB) (red) with respect to the transcription rate. The Hill coefficient, h, was fixed at 3 for the simulations. The initial conditions for both (a) and (b) were $r_1 = 10$, $r_2 = 2$, $r_3 = 3$, $p_1 = 5$, $p_2 = 1$, and $p_3 = 6$. **Code written to create this figure appears in Appendix A, Section A.1.2.**



(a)



(b)

Figure 3.2: (a) Period of the concentration of protein 1 for models (SS) (black) and (SB) (red) with respect to the Hill coefficient. Similar to the amplitude comparison in Figure 3.1(a), the transcription rates were set to be equivalent at $k = 10$. (b) Period of the concentration of protein 1 for models (SS) (blue) and (SB) (red) with respect to the transcription rate. The Hill coefficient, h, was fixed at 3 for the simulations. The initial conditions for both (a) and (b) were $r_1 = 10$, $r_2 = 2$, $r_3 = 3$, $p_1 = 5$, $p_2 = 1$, and $p_3 = 6$. **Code written to create this figure appears in Appendix A, Section A.1.2.**

at every two-tenth value. We numerically solve the systems to either the steady state or the limit cycle. To compute the period, we perform an event location procedure. The procedure first finds a time point when $p_1 = p$ and $\left.\frac{dp_1}{dt}\right|_{p_1=p} > 0$, where p is a concentration known to be in the limit cycle. Then, the algorithm finds the next time point in which $p_1 = p$ and $\left.\frac{dp_1}{dt}\right|_{p_1=p} > 0$ and saves this time point. The period is then estimated to be the difference between the two time points.

Figure 3.2(a) shows the periods of the two models with respect to the Hill coefficient. Interestingly, the period of model (SB) increases more rapidly with respect to h and eventually surpasses the period of model (SS) ($h \approx 4.75$, Figure 3.2(a)). We hypothesize this phenomenon is a result of the increased time delay present in the successive-binding transcription-rate function. Because we incorporate successive binding of the repressor proteins to the promoter, the time required to repress transcription increases, resulting in an elongated period of model (SB).

Next, we fix the Hill coefficient, $h = 3$, and let the transcription rates vary. Again, we set $k_1 = k_2 = k_3 = k$ and vary k from 3 to 50. Figure 3.2(b) shows, for both models, the periods of the first protein concentration with respect to the transcription rate. In Figure 3.2(b), we see that the variation in the period with respect to the transcription rate does not differ significantly between the two models. This finding is not surprising as an increase in the Hill coefficient contributes more time delay in model (SB) than an increase in the transcription rate.

3.3.3 Phase

Finally, we analyze how the phase differs between models (SS) and (SB). Again, we fix all parameters except the Hill coefficient and let h vary from 2 to 50. To estimate the phase, we perform an event location procedure that computes the peak of the abundances of proteins 1 and 2. We find the peak of protein 1 by having the ODE solver find when $\dot{p}_1 = 0$ and decreasing (similarly for protein 2). After computing the peaks of the protein abundances within the limit cycle, we calculate the phase difference by subtracting the time of protein 2 abundance from the time of protein 1 abundance. The phase corresponds to the strength of the repression of protein 1 on the transcription of gene 2. Thus, a longer time between the two peaks indicates that protein 1 has more of a repressive action than a lesser phase difference.

Figure 3.3(a) shows the phase difference in protein 1 and protein 2 abundance levels between the two models with respect to the Hill coefficient. Similar to the period comparison in Figure 3.2(a), the Hill coefficient does not affect phase difference for model (SS) (transcription rate given by the Hill function). In contrast, the phase difference of model (SB) continues to increase with increasing Hill coefficient (red curve in Figure 3.3(a)). This phenomenon is due to the strength of repression by protein 1 on protein 2 transcription increasing more rapidly given the successive-binding transcription rate.

Figure 3.3(b) shows the phase difference of protein 1 and protein 2 abundances between the two models with respect to the transcription rate. It is interesting that both increase with increasing transcription rate. This phenomenon may appear counterintuitive initially because, with an increasing transcription rate, one would expect the repressive effect of protein 1 to decrease. However, with a larger transcription rate, more proteins are produced through translation, which contributes to more repression of the next protein's production. Unlike when varying the Hill coefficient, however, the phase differences between the two models increase at similar rates.

3.4 Analytical Comparison of Amplitudes

In light of our numerical comparison of the amplitudes in Section 4.5.4, in this section, we pursue a theoretical result proving that, all parameters being equal, a repressilator model with the successive-binding transcription-rate function will exhibit smaller amplitudes than a repressilator model in which transcription is modeled by a Hill function. To our knowledge, however, there is limited information on how to compute rigorously amplitudes of limit-cycle oscillations, even for the simplest of ODE systems. Given the state of the field in this respect, we obtain only partial results, and then we give a formal conjecture at the end of the section.

First, we consider a repressilator model of the form:

$$\begin{aligned} \dot{r}_i &= \frac{\alpha_i}{1 + p_{i-1}^{h_i}} - d_{r_i} r_i, \\ \dot{p}_i &= k_i r_i - d_{p_i} p_i, \end{aligned} \tag{SS-2}$$

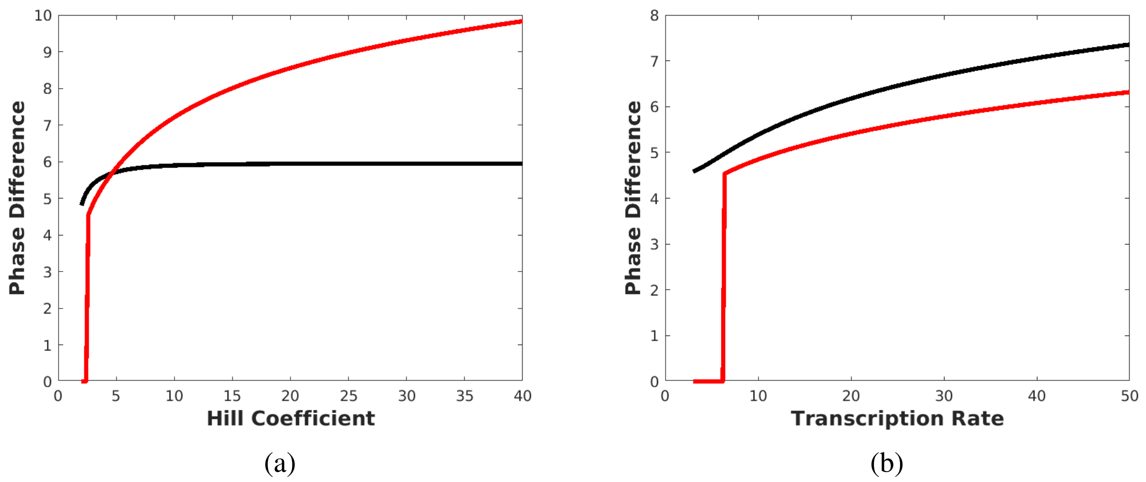


Figure 3.3: (a) Phase difference (in time) of the abundances of proteins 1 and 2 for models (SS) (black) and (SB) (red) with respect to the Hill coefficient. Similar to the amplitude and period comparisons in Figures 3.1(a) and 3.2(a), the transcription rates were set to be equivalent at $k = 10$. (b) Phase difference (in time) of the abundances of proteins 1 and 2 for models (SS) (black) and (SB) (red) with respect to the transcription rate. The Hill coefficient, h, was fixed at 3 for the simulations. The initial conditions for both (a) and (b) were $r_1 = 10$, $r_2 = 2$, $r_3 = 3$, $p_1 = 5$, $p_2 = 1$, and $p_3 = 6$. **Code written to create this figure appears in Appendix A, Section A.1.2.**

for $i = 1, \dots, n$. Here, the parameters α_i are the transcription rates, the k_i's the translation rates, d_{r_i} (d_{p_i}) the degradation rate of mRNA-i (protein-i, respectively), and the h_i's the Hill coefficients. Note that system (SS-2) generalizes system (SS) to allow for general degradation, transcription, and translation rates.

Similarly, we generalize system (SB) to allow for general degradation, transcription, and translation rates. Consider a repressilator system with the transcription-rate function given by the newly derived function in Section 3.2

$$\begin{aligned} \dot{r}_i &= \frac{\alpha_i}{(1 + p_{i-1})^{h_i}} - d_{r_i} r_i, \\ \dot{p}_i &= k_i r_i - d_{p_i} p_i, \end{aligned} \tag{SB-2}$$

for $i = 1, \dots, n$.

For these two systems, we prove a theorem that confines possible limit cycles to a compact region in $\mathbb{R}^{2n}_{\geq 0}$, where n is the number of genes in the system.

Theorem 3.4.1. Consider system (SS-2) with a limit cycle $\gamma \subset \mathbb{R}^{2n}_{\geq 0}$ and period T. Let $\Pi_i : \gamma \to \mathbb{R}^2_{\geq 0}$ where $\Pi_i(x) := (r_i, p_i)$ where $x = (r_1, r_2, \ldots, r_n, p_1, p_2, \ldots, p_n)$. Then $\Pi_i(\gamma) \subset \Gamma_i$ where

$$\Gamma_i = \left[0, \frac{\alpha_i}{d_{r_i}}\right] \times \left[0, \frac{k_i \alpha_i}{d_{r_i} d_{p_i}}\right].$$

Proof. From the equation for $\dot{r}_i$ in system (SS-2), we have

$$\dot{r}_i = \frac{\alpha_i}{1 + p_{i-1}^{h_i}} - d_{r_i} r_i < \alpha_i - d_{r_i} r_i. \tag{3.15}$$

Let $\bar{x} = (\bar{r}_1, \ldots, \bar{r}_n, \bar{p}_1, \ldots, \bar{p}_n)$ with $\bar{r}_i > \dfrac{\alpha_i}{d_{r_i}}$. Then, by Eqn. (3.15), we have

$$\left.\frac{dr_i}{dt}\right|_{\bar{x}} < \alpha_i - d_{r_i} \bar{r}_i < 0.$$

Since $\dot{r}_i < 0$ at all points $\bar{x}$ with $\bar{r}_i > \dfrac{\alpha_i}{d_{r_i}}$, the point $\bar{x}$ cannot be in the limit cycle γ. For, if $\bar{x}$ were in the limit cycle, then the limit cycle must return to $\bar{x}$, but this is impossible due to $\dfrac{dr_i}{dt} < 0$ for all points in some neighborhood of $\bar{x}$. Therefore, if $x^* = (r_1^*, \ldots, r_n^*, p_1^*, \ldots, p_n^*)$ is in the limit cycle γ, then $r_i^* \leq \dfrac{\alpha_i}{d_{r_i}}$.

The argument for showing that

$$p_i \leq \frac{\alpha_i k_i}{d_{r_i} d_{p_i}}$$

follows similarly plus we use the fact that $r_i \leq \dfrac{\alpha_i}{d_{r_i}}$.

Our choice of i was arbitrary, so, for all $i = 1, \ldots, n$, the projection $\Pi_i(\gamma) \subset \Gamma_i$ where

$$\Gamma_i = \left[0, \frac{\alpha_i}{d_{r_i}}\right] \times \left[0, \frac{k_i \alpha_i}{d_{r_i} d_{p_i}}\right].$$

$\square$

The analogous result holds for system (SB-2).

Theorem 3.4.2. Consider system (SB-2) with n genes with a limit cycle $\gamma \subset \mathbb{R}^{2n}_{\geq 0}$ and period T.

Let $\Pi_i : \gamma \to \mathbb{R}^2_{\geq 0}$ where $\Pi_i(x) = (r_i, p_i)$ for $x \in \gamma$. Then $\Pi_i(\gamma) \subset \Gamma_i$ where

$$\Gamma_i = \left[0, \frac{\alpha_i}{d_{r_i}}\right] \times \left[0, \frac{k_i \alpha_i}{d_{r_i} d_{p_i}}\right].$$

Proof. The proof is analogous to the proof of Theorem 3.4.1. □

Theorems 3.4.1 and 3.4.2 are significant *per se* because they give an upper bound on the limit-cycle amplitude of systems (SS-2) and (SB-2), respectively, with respect to the parameters describing the rates of transcription, translation, and degradation of the i-th gene and protein. To illustrate the upper bounds in Theorems 3.4.1 and 3.4.2, in Figure 3.4, we extend Figure 3.1(a) to include a line through the origin with slope given by the quantity $\dfrac{\alpha_i k_i}{d_{p_i} d_{r_i}}$. For systems (SS) and (SB), the protein bound given in Theorems 3.4.1 3.4.2 is equivalent to the transcription rate, α_i. In Figure 3.4, notice that the amplitude of protein 1 for the model (SS) (black line) closely matches the upper bounds from Theorems 3.4.1 and 3.4.2 whereas the amplitude of protein 1 of model (SB) (red line) does not.

Next, we investigate how the limit cycles compare between models with the Hill function versus the successive-binding transcription-rate function. In particular, we ask whether the limit cycle of the model with the successive-binding transcription-rate function is always "embedded" in the limit cycle of a model with the Hill function. To clarify what we mean by "embedded", we view the limit cycle γ as a *Jordan curve*–a plane curve which is topologically equivalent to the unit circle, i.e., it is simple and closed [54]. Then, by the Jordan curve theorem, we divide the complement of the limit cycle into the interior and exterior [54]. We denote the interior of the limit cycle γ as $int(\gamma)$. To pose our question rigorously, we ask:

Question 3.4.3. Let γ_1 be the limit cycle of a model with the successive-binding transcription-rate function and γ_2 the limit cycle of the corresponding model with the Hill function. Then, is it the case that the graph of $\Pi_i(\gamma_1) \subset int(\Pi_i(\gamma_2))$?

The answer to Question 3.4.3 is no. Essentially, the strength of repression given by the successive-binding transcription-rate function allows for the minimum protein levels to drop below

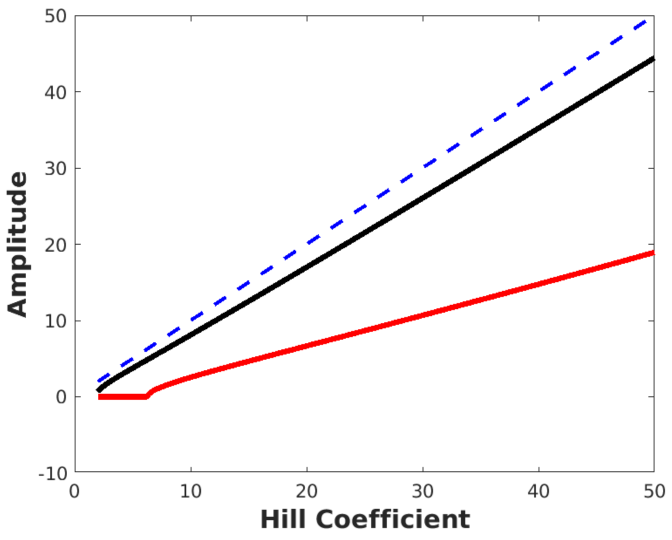
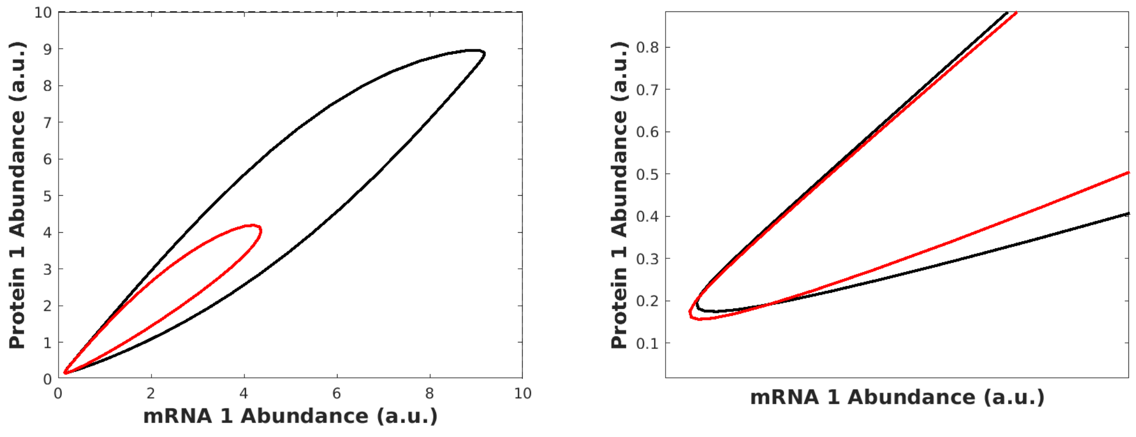


Figure 3.4: We obtain this figure by starting with Figure 3.1(b). The dotted blue line is the bound, from Theorem 3.4.1, for the amplitude of protein-1 abundance. In the case of system (SS), the bound is $p_i \leq \dfrac{\alpha k}{d_{r_i} d_{p_i}} = \alpha$, where α is the transcription rate.

the minimum protein levels of a model with the Hill function. Figure 3.5(a) plots the projection Π_i to $\mathbb{R}^2_{\geq 0}$ of the limit cycle of a model with the Hill function (black) and the limit cycle of the corresponding model with transcription-rate function given by the successive-binding function. Figure 3.5(b) zooms in on the trough of the limit cycles to show that, in fact, the minimum protein 1 (and mRNA 1) abundance for the model with the successive-binding function is smaller than that of the model with the Hill function.

Finally, returning to Question 3.4.3, we conclude the section with a conjecture. To state the conjecture, consider generalized repressilator models of the form

$$\begin{aligned} \dot{r}_i &= \frac{\alpha_i}{K_i + p_{i-1}^{h_i}} - d_{r_i}(r_i), \\ \dot{p}_i &= k_i(r_i) - d_{p_i}(p_i). \end{aligned} \tag{SS-3}$$



(a) (b)

Figure 3.5: (a) The black curve is the projection $\Pi_1(r_1, \ldots, r_n, p_1, \ldots, p_n) = (r_1, p_1)$ to $\mathbb{R}^2_{\geq 0}$ of the limit cycle of a repressilator system with transcription modeled by the Hill function. The red curve is the same projection to $\mathbb{R}^2_{\geq 0}$ of the limit cycle of a repressilator system with transcription modeled by the successive-binding function. (b) Zooming in on the trough of the limit cycles plotted in (a). **The code we wrote to generate this figure appears in Appendix A, Section A.1.3**.

and

$$\begin{aligned} \dot{r}_i &= \frac{\alpha_i}{(K_i + p_{i-1})^{h_i}} - d_{r_i}(r_i), \\ \dot{p}_i &= k_i(r_i) - d_{p_i}(p_i). \end{aligned} \tag{SB-3}$$

Conjecture 3.4.4. Consider generalized models of the repressilator (SS-3) and (SB-3) where the functions d_{r_i}, $k_i(r_i)$, and d_{p_i}, respectively, are equivalent between the two models. Then the amplitude of r_i for system (SS-3) is greater than the amplitude of r_i for system (SB-3). Likewise, the amplitude of p_i for system (SS-3) is greater than the amplitude of p_i for system (SB-3).

3.5 Discussion

Currently, the Hill function is the standard function used to model the process of transcription in systems of gene regulatory networks [45]. However, even a preliminary theoretical result regarding the Hill function–Griffith's result that a Hill coefficient of 8 is required for oscillations–yielded a biologically unreasonable conclusion. Therefore, we derived a new transcription-rate function labeled the *successive-binding function* that arose from more reasonable biological assumptions.

In addition, we showed that incorporating the new successive-binding function within a model of the repressilator leads to significant changes in dynamics. For example, numerical simulations showed that amplitudes, periods, and phase differences between proteins of a model constructed with the old transcription-rate function differed significantly from those of a model with our new function. Numerical simulations revealed that, with the successive-binding function, the period was more sensitive to the Hill coefficient and continued to increase with increasing Hill coefficient whereas the period stabilized quickly with respect to the Hill coefficient given the model using the Hill function. We found that the phase difference behaved similarly to the period. Protein abundance amplitudes, however, stabilized quickly for a model with the new transcription-rate function as opposed to those computed from a model with the Hill function as the transcription-rate function. These differences in dynamics are a result of the new transcription-rate function modeling a stronger repressive action than the previous Hill function.

Furthermore, we worked towards a theoretical result confirming our numerical findings that the amplitudes of proteins from a model with the new successive-binding function are smaller than those from a corresponding model with the Hill function. We proved that limit cycles of both models, when projected to $\mathbb{R}^2_{\geq 0}$, are confined to a particular region that is defined by the parameters of the model. We showed numerically that the amplitudes of proteins of the model with the Hill function are close to the bound of this region whereas the amplitudes of the other model do not. In the end, we conjecture a formal result comparing the amplitudes of the two models. However, the current state of the field of dynamical systems has limited results on explicitly computing amplitudes of limit cycles. Thus, a future direction of research is to explicitly compute amplitudes of the two repressilator models in terms of the parameters and, more generally, of systems of ODEs that exhibit limit-cycle oscillations.

4. NOVEL ALGORITHMS FOR ESTIMATING PARAMETERS OF THE REPRESSILATOR

4.1 Introduction

Historically, scientists have addressed two problems regarding parameters of the repressilator: identifying parameter regions yielding certain dynamics (e.g., [55, 56]) and *parameter estimation*–determining unknown system parameters from measurements of other quantities (e.g., [57, 6]). These two issues are related to the two main motivations of the field of synthetic biology, discussed previously in Section 2: engineering certain cellular behaviors and gaining a deeper functional understanding of natural biological systems. For example, in the interest of bioengineering and synthetic biology (motivation 1 mentioned above), Strelkowa and Barahona studied analytic conditions for the emergence of a finite sequence of periodic orbits leading to reachable long-lived oscillating transients [55].

Below, we are concerned with the latter problem: estimating model parameters of the repressilator given experimental data. Parameter estimation is an important problem in system biology because it is a crucial step in obtaining predictions from computational models of biological systems. Therefore, our main motivation for the study is a better understanding of naturally occurring biological networks and systems using a more efficient and faster parameter estimation procedure than current algorithms. An important future direction of our work is–with a more robust parameter estimation procedure than those currently used–better investigating the variation in the rates at which the processes of transcription, translation, and degradation of the repressilator genes and proteins occur within cells. Moreover, accurately estimating the Hill coefficient parameter, in light of our discussion in Remark 3.1.1, will lead to biological insights, as the coefficient relates to the number of proteins that bind to a gene promoter.

Furthermore, with a parameter estimation procedure that generates accurate fits, we can address the issue of model selection. Recall that, in Section 2, we generalized a well-known model of the repressilator by Müller *et al.* Specifically, in our generalization, we allowed for monotone functions to model the processes of transcription, translation, and degradation. With a generalized model and a more accurate parameter estimation algorithm, we can compare model solutions that arise from two differing repressilator systems. For example, in future work, with our new algorithm, we will investigate the faithfulness of solutions of the repressilator with the Hill function modeling transcription (see Remark 2.3.1 in Section 2) to solutions of the repressilator with our newly-derived transcription-rate function from Section 3.

Along with a better functional understanding of biological systems, another motivation of our work is the need for new techniques that address the problems of parameter estimation for biological oscillators in general. Estimating parameters of models of biological oscillators is inherently challenging due to the nonlinearity present in the system. In particular, algorithms must overcome two key challenges: *multimodality*–the presence of many local optima in the objective-function landscape [58]–and the lack of all initial data. In Section 4.3, we illustrate how these two issues complicate parameter estimation of the repressilator.

Recently, to address multimodality, a new toolbox called GEARS was developed to perform global optimization of parameters of nonlinear systems [6]. The toolbox was applied to four well-known biological oscillators, one in particular being the repressilator. As long as we input all initial conditions, the procedure works quickly and generates accurate parameter estimates [6]. However, when there are unknown initial conditions, the algorithm fails to identify accurate parameter estimates (see Section 4.4.2). Thus, the procedure fails to address the challenge two of parameter estimation of the repressilator, the lack of initial mRNA data.

To address the issue of lack of initial mRNA values, in Section 4.6, we introduce an algorithm taking as input a parameter estimate and discrete data and outputs model solution values at time points of the discrete data set. The procedure works independently of knowing all initial data points and extracts model solution values inside the limit cycle. Therefore, the new algorithm can be coupled with global optimization procedures, such as GEARS, to address challenge two, the lack of initial mRNA data. An important future iteration of the GEARS Toolbox will be to incorporate a procedure similar to our new algorithm to handle parameter estimation of models where some initial data is unknown.

Another commonly used parameter estimation procedure in computational biology is the Constrained Hybrid Extended Kalman Filter (HEKF) Algorithm with an *a posteriori identifiability test*–a test that assesses the reliability of the parameter estimate [57]–using the variance, called the *Variance Test*. Using simulated data from a repressilator model, Khammash and Lillacci employed the Constrained HEKF Algorithm with the Variance Test to generate parameter estimates. When the repressilator model was run with the parameter estimate output from the algorithm, the solution closely matched the original model solution [57]. However, Khammash and Lillacci did not report the original parameters used to simulate the data nor the parameter estimates generated by the algorithm. In contrast, in our implementations of the Constrained HEKF Algorithm with Variance Test, we see inaccuracies in the three parameters, with the most variation in the Hill coefficient. In Section 4.4.1, we show the Variance Test sometimes rejects parameter estimates more accurate than those it accepts.

As a discriminator among possible parameter estimates, the variance is inadequate because it fails to incorporate any information regarding the oscillatory nature of the repressilator. Therefore, to replace the variance as a *test statistic*–a quantity to discriminate among possible parameter estimates–in the Constrained HEKF Algorithm *a posteriori* identifiability test, in Section 4.5, we investigate how certain quantities related to biological oscillators vary with respect to the parameters of the repressilator system. In particular, we seek a quantity, or quantities, that are sensitive to variations in the parameters. Example quantities that we analyze include the period of oscillations, the amplitude of protein abundance levels, the skewness (see Section 4.2.2), the cost of protein production [59] (see Section 4.2.3), and others.

In our numerical analysis, we find that the test statistics we considered, in general, are sensitive to variations in the transcription rate and degradation ratio but rarely sensitive to variations in the Hill coefficient parameter. In fact, most quantities are initially sensitive to the Hill coefficient but stabilize soon after. Ultimately, in our first algorithm, we incorporate the period, the amplitude, and the cost of protein production (see Section 4.2.3) into an identifiability test coupled with the

Constrained HEKF Algorithm to discriminate among possible parameter estimates. The algorithm takes as input the discrete, time-course data of protein abundance levels and outputs the parameter estimate. In addition, we introduce a second algorithm that discriminates among parameter estimates using the ℓ^2-norm between the data points and the model solution with the parameter estimate (see Section 4.6).

For an initial investigation into the effectiveness of the two algorithms, we simulate data sets with various spacings and noise added. We show that, after inputing these data sets, the two algorithms produce accurate parameter estimates of the repressilator. The second algorithm, however, generates parameter estimates consistently more accurate than those of the first algorithm. We continue, in Section 4.8.1, to compare the effectiveness of the second algorithm to that of other commonly used algorithms in computational biology. In summary, we show that our new algorithm is as accurate and faster than existing algorithms.

4.2 Background

Here, we present background for the subsequent analysis. In particular, we outline the Constrained HEKF algorithm, which was first applied to the repressilator in [57]. As a substitute to the Variance Test used in [57], we propose several test statistics. In particular, we consider the skewness as one possible test statistic, so we define our version of the skewness of a distribution. Next, we recall a quantity introduced by Jo *et al.* in [59], namely the cost of protein production. Finally, since we consider the period as a test statistic, we outline how we use the Fourier transform to estimate the period from a discrete data set.

4.2.1 The Constrained HEKF Algorithm

We give a brief overview of the Constrained HEKF algorithm presented by Khammash and Lillacci in [57]. For more details on the derivation and methods, see [57] and [60].

4.2.1.1 Parameter Estimation

We begin with a model of the form

$$\begin{cases} \dot{x} = f(x, u, \theta) \\ \dot{\theta} = 0 \\ x(t_0) = x_0 \\ \theta(t_0) = \theta_0 \\ y = h(x). \end{cases} \tag{4.1}$$

The state vector x contains the concentrations of the species of interest; the vector θ contains the parameters to be estimated by the algorithm; the output vector y represents the measurements, which are functions of the state vector x.

First, the algorithm converts the problem into one of *state extension*–considering parameters as states of the system and estimating the initial conditions that generate the observed output y. The algorithm proceeds in two main steps: (1) The *prediction step* where the estimates are updated based on the model (*a priori estimate*) and (2) The *correction step* where the estimates are updated based on the measurements (called the *a posteriori estimate*). At each time point, new estimates are computed based on the previous *a posteriori* estimates.

To make these steps more precise, we introduce some notation. First, consider the system

$$\begin{cases} \dot{x} = f(x, u, \theta) + w \\ y_k = h(x_k(t_k)) + v_k. \end{cases} \tag{4.2}$$

From (4.2), notice that we retain the continuous ODE model for the state vector x but allow for discrete measurements of the outputs, y_k. Assume that we have measurements y_k for times $t_1, ..., t_s$. Then the algorithm computes *a priori* and *a posteriori* estimates corresponding to each time point. We follow the notation in [57] and denote the *a priori* estimate at time t_k as x_k^- and the *a posteriori* estimate at time t_k as x_k^+. The algorithm also computes estimates for the *error covariance matrix*, P, at each time step. The *a priori* error covariance estimate, P_k^-, is formed by integrating a differential Lyapunov function using the previous *a posteriori* error covariance matrix, P_k^+, as an initial condition.

The variable w is called the *process noise* and inversely represents the confidence in the *model*. The process noise is assumed to be Gaussian with mean zero and with a covariance matrix Q. The matrix Q is an $n \times n$ matrix where n is the number of states in the system. The variable v_k is called the *measurement noise* and, similarly, inversely represents confidence in the measurements. The measurement noise is also assumed to be Gaussian with mean zero, and we denote its covariance matrix by R. The matrix R is a $p \times p$ matrix, where p denotes the number of measurements (length of the vector y).

With the notation in place, we outline the main steps of the algorithm [57]. Code for implementing this algorithm for parameter estimation of a simple repressilator system appears in Appendix A, Section A.2.1.

Algorithm 4.2.1. Constrained HEKF algorithm summary [57]

	Models (4.1) and (4.2).
INPUT:	Time points $t_1, \dots, t_s$.
	Measurements $y_1^{(i)}, \dots, y_s^{(i)}$ for $i = 1, \dots, p$.
OUTPUT:	Parameter estimate $\hat{\theta}$.

1. Initialize the system.

2. Compute the Jacobians of f and h at the previous *a posteriori* estimate.

$$A_k = \frac{\partial f}{\partial x}|_{x=x_{k-1}^+}, \quad H_k = \frac{\partial h}{\partial x}|_{x=x_{k-1}^+}$$

3. Advance to the next time step.

$$\begin{cases} \dot{x} = f(x,u) \\ x(t_{k-1}) = x_{k-1}^{+} \end{cases} \implies x_k^- = x(t_k).$$

$$\begin{cases} \dot{P} = A_k P + P A_k^T + Q \\ P(t_{k-1}) = P_{k-1}^{+} \end{cases} \implies P_k^- = P(t_k).$$

4. Compute the gain.

$$L_k = P_k^- H_k^T (H_k P_k^- H_k^T + R)^{-1}$$

5. Incorporate the current measurement to correct the prediction step.

$$x_k^+ = x_k^- + L_k(y_k - h_k(x_k^-))$$

$$P_k^+ = (I - L_k H_k) P_k^- (I - L_k H_k)^T + L_k R_k L_k^T$$

6. Check if the estimates satisfy constraints on the states and parameters. If not, optimize

$$\hat{x}_{k+1}^{+} = \text{arg} \quad \min (x_{k+1} - x^+k + 1)^T (P_k^+)^{-1} (x_{k+1} - x_{k+1}^+),$$

subject to the linear constraints $Dx_{k+1} \leq d_{k+1}$.

7. Repeat steps 2-6 for each time point $t_1, ..., t_s$.

8. Output θ_s^+.

4.2.1.2 Variance Test

Along with the algorithm to estimate the parameters as states, Khammash and Lillacci gave a test to accept or reject the estimates with a user-defined level of confidence $\gamma \in (0,1)$. The test is based on a simple estimation of the variance of a random variable. To outline the test, recall that s

denotes the number of time points, and p denotes the number of quantities being measured. Also, R is a $p \times p$ matrix with diagonal entries σ_i^2 corresponding to the variance of each measurement for $i = 1, ..., p$. Then the steps of the variance test are as follows.

Algorithm 4.2.2. The Variance Test [57]

INPUT:	Parameter estimate $\hat{\theta}_0$. Confidence γ. Measurements $y_1^{(i)}, \ldots, y_s^{(i)}$ for $i = 1, \ldots, p$.
OUTPUT:	Accept or reject $\hat{\theta}_0$ with confidence γ.

1. Compute the parameter estimates, $\hat{\theta}_0$, based on the algorithm in Section 4.2.1.1.

2. For each measurement $i = 1, ..., p$, compute

$$\hat{v}_k^{(i)} = y_k^{(i)} - h_k^{(i)}(x_{\hat{\theta}_0}(t_k)).$$

   This gives s samples of a Gaussian random variable assumed to have zero mean.

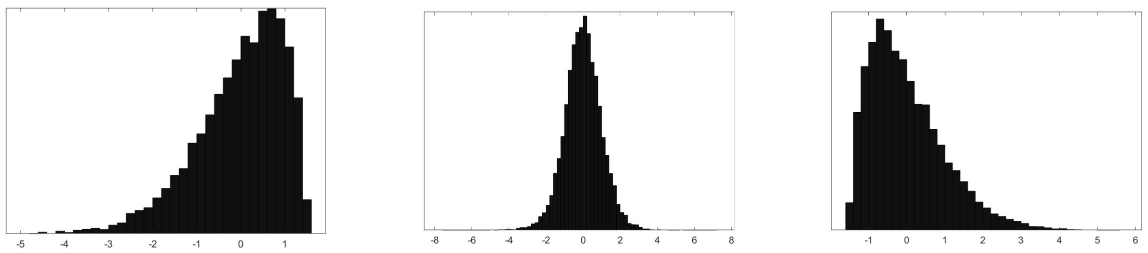
3. Compute the variance for $i = 1, ..., p$:

$$\hat{\sigma}_i^2 = \frac{1}{s}\sum_{k=1}^{s}(\hat{v}_k^{(i)})^2.$$

4. For confidence $\gamma = 1 - \delta$, compute confidence intervals of the variance for $i = 1, ..., p$:

$$\left[\frac{s\hat{\sigma}_i^2}{\chi_{s,1-\delta/2}}, \frac{s\hat{\sigma}_i^2}{\chi_{s,\delta/2}}\right].$$

5. Check whether σ_i^2 is in the above interval for all $i = 1, ..., p$. If σ_i^2 is outside the interval for some i, **REJECT** the parameter estimates with a confidence γ. Else, **ACCEPT** the parameter estimate with a confidence γ.



(a) (b) (c)

Figure 4.1: Sample distributions, all with equivalent mean (0), standard deviation (1), and fourth-moment, kurtosis (4). Each distribution was found in MATLAB with the function *pearsrnd* [4], which allows you to specify a mean, standard deviation, skewness, and kurtosis. There are 10,000 data points in each distribution. (a) Distribution with skewness equal to -1. (b) Distribution with skewness equal to 0. (c) Distribution with skewness equal to 1.

4.2.2 Skewness

The mean (the first moment) and the variance (second moment) provide information on the central location and the spread about that location, respectively, of a given data set [61]. The third moment, called *skewness*, is

$$\frac{1}{N}\sum_{i=1}^{N}\left[\frac{x_i-\bar{x}}{\sigma}\right]^3, \tag{4.3}$$

where σ is the standard deviation of the data set [61].

The skewness of a waveform provides information on the bias of the tail of the distribution to one direction or the other. For example, a negative skewness of a distribution corresponds to a distribution with a longer tail to the left than to the right (Figure 4.1(a)). A distribution with positive skewness will have a longer tail to the right than to the left (Figure 4.1(c)). Finally, a distribution with zero skewness will have tails centered around the mean (Figure 4.1(b)). As this quantity is intimately linked to shapes of waveforms, we consider it as a way to identify parameter sets of models that exhibit oscillatory behavior more accurately than with the variance as done in Section 4.2.1.2.

4.2.3 The Cost of Protein Production

In [59], Jo *et al.* investigate the relationship between waveform shapes of oscillations and the underlying biochemical process. They derive a mathematical framework that uses waveforms to reveal previously hidden biochemical mechanisms of circadian timekeeping. Since parameter estimation is intimately linked with matching shapes of oscillations, we apply portions of their framework to our parameter estimation problem.

First, we introduce the type of differential equation that Jo *et al.* consider. Let $x(t)$ denote the production of a protein over time. For many systems, the dynamics of $x(t)$ are described by an equation of the form

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = g(t) - r(t)x(t). \tag{4.4}$$

The functions $g(t)$ and $r(t)$ correspond to protein synthesis and protein degradation over time, respectively.

Remark 4.2.3. Equations of the form (4.4) are common in many models for circadian clocks. In fact, Forger and Kim considered a slightly more general type of equation in [62] and then proved results about existence and uniqueness of the functions $g(t)$ and $r(t)$, given perfect or noisy time-course data. Also, the equations governing protein production in the repressilator system are of the form (4.4) [5, 20], where $r(t)$ is assumed to be a constant [5, 20].

The following definition was introduced in [59]:

Definition 4.2.4. The *cost of protein production*, c, of a protein with dynamics governed by (4.4) is:

$$c := \frac{\Delta x}{T} = \langle g(t) \rangle = \langle r(t)x(t) \rangle, \tag{4.5}$$

where Δx denotes the amount of protein synthesized over the period T, and $\langle \cdot \rangle$ represents the time average from time 0 to T, i.e.,

$$\langle g(t) \rangle := \frac{1}{T}\int_0^T g(t)dt.$$

Here, we propose the cost, c, as a test statistic for a few reasons. First, the cost has been shown to be linked to the waveform of oscillations [59]. Second, the cost c can be approximated from discrete protein levels using, for example, the function *trapz* in MATLAB. Finally, for the repressilator, experimental data sets rarely include both mRNA and protein expression time courses. Thus, we need a quantity that can be approximated from only one type of profile. From Eqn. (4.5), we see that protein levels are sufficient to compute the cost.

4.2.4 Estimating the Period of a Discrete Data Set Using the Fast Fourier Transform

Finally, we propose the period of oscillations as another test statistic to discriminate among Hill coefficients. However, the period can be hard to approximate given a discrete data set. So, for our analysis, we use the following procedure to estimate the period. We assume that there are k time points, $t_1, \ldots, t_k$ that are uniformly spaced with a spacing of τ.

1. Compute the power spectrum of the data set using the Fast Fourier Transform (FFT).
2. Compute ω values from the time points as
$$\omega_i = \frac{2\pi t_i}{t_k}, \quad \text{for} \quad i = 1, \ldots, k. \tag{4.6}$$
3. Plot the power spectrum with respect to the ω values.
4. Find the ω value that gives the peak of the power spectrum. Call it ω^*.
5. The estimated period is
$$T = \frac{2\pi}{\omega^*}. \tag{4.7}$$

There are issues with estimating the period in this way. First, let $\hat{\omega}$ denote the actual frequency of the model. Then, from (4.6), we know that

$$\frac{2\pi t_i}{t_k} \leq \hat{\omega} \leq \frac{2\pi(t_i + \tau)}{t_k}, \tag{4.8}$$

for some $i = 1, \ldots, k-1$. From (4.8), the estimated frequency differs from the actual frequency by as much as $\frac{2\pi\tau}{t_k}$. Say, for example, that a circadian experiment gathers data every 4 hours for two days. Then, the maximum possible error in the estimated frequency is

$$\frac{8\pi}{48} = \frac{\pi}{6}.$$

Therefore, care should be taken for data to be gathered for as long as possible at the smallest possible interval if one wants to estimate the period.

4.3 Challenges in Parameter Estimation of the Repressilator

Parameter estimation of the repressilator, as with other biological oscillators, is challenging due to the nonlinearity present in the system. Here, we highlight two such challenges: multimodality and the lack of initial values for mRNA abundances. We illustrate these issues by generating sample data from a repressilator system and then attempting to estimate the parameters using traditional techniques.

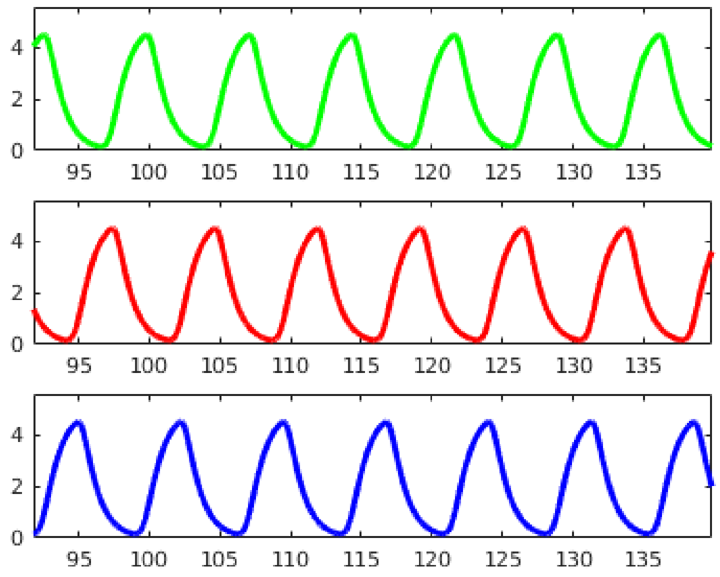


Figure 4.2: Simulated protein abundances ($p_1(t)$, $p_2(t)$, and $p_3(t)$, respectively) arising from system (4.9) with initial conditions $r_1 = 3$, $r_2 = 6$, $r_3 = 10$, $p_1 = 10$, $p_2 = 2$, and $p_3 = 10$.

Variable/Parameter	Description
r_i	abundance of mRNA-i
p_i	abundance of protein-i
α_0	rate of transcription that occurs in the presence of saturating amounts of repressor [5]
α	transcription rate
β	ratio of the degradation rate of proteins to the degradation rate of mRNAs
n	Hill coefficient

Table 4.1: A list of the variables and parameters of system (4.9).

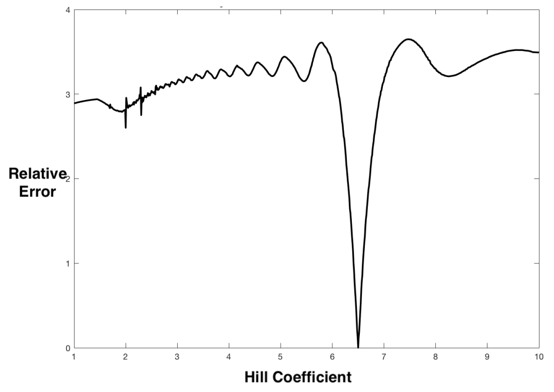
4.3.1 Simulated Repressilator Data

To begin, recall, from Section 2.2.1, the three-gene repressilator system from [5]:

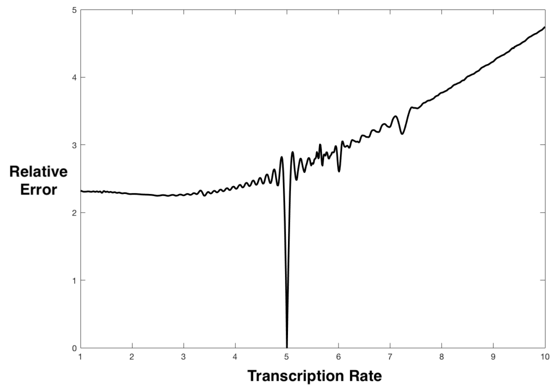
$$\begin{aligned} \dot{r}_i &= \alpha_0 + \frac{\alpha}{1 + p_{i-1}^n} - r_i, \\ \dot{p}_i &= \beta(r_i - p_i), \end{aligned} \tag{4.9}$$

where i =1, 2, and 3. The parameter α is the transcription rate for each gene. The parameter β is the ratio of the degradation rate of the protein to the degradation rate of the mRNA. The parameter n is the Hill coefficient. Finally, the parameter α_0 is the rate of transcription that occurs in the presence of saturating amounts of repressor [5]. Hereafter, we use system (4.9) for all subsequent analysis and simulations, so, in Table 4.1, we list the variables and parameters for reference.

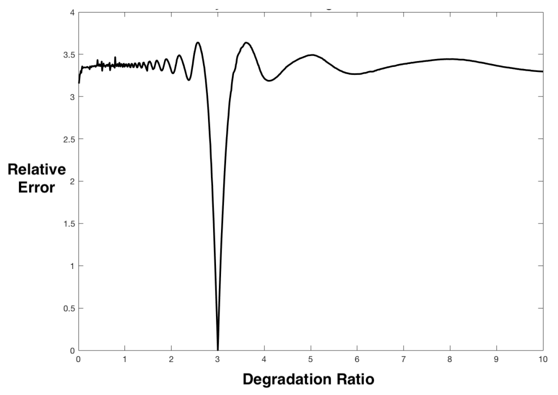
To illustrate challenges of fitting parameters of the repressilator, we simulate data from system (4.9) with the parameters $\alpha = 5$, $\beta = 3$, $n = 6.5$, and $\alpha_0 = 0$. Figure 4.2 plots the protein abundances we generated for system (4.9). The initial conditions, given in the caption of Figure 4.2, were randomly generated integers between 1 and 10. We display the protein abundances starting at time $t = 90$ to ensure that the system is in the limit cycle (Figure 4.2).



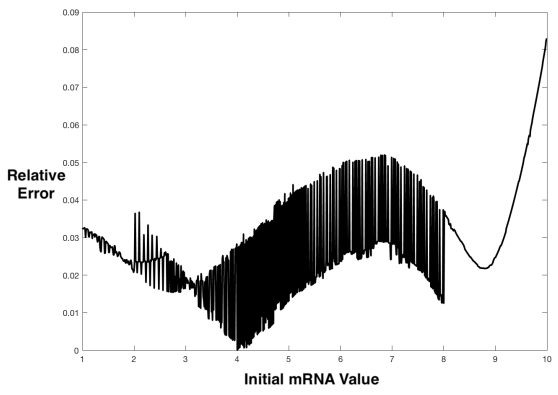
(a) Relative error as a function of the Hill coefficient.



(c) Relative error as a function of the transcription rate.



(b) Relative error as a function of the degradation ratio.



(d) Relative error as a function of one initial mRNA value.

Figure 4.3: Objective plots with respect to each parameter value based on system (4.9). See Section 4.3.2 for a description of how we calculate the objective function (Eqn. (4.10)). For each simulation, the parameters that were not varied were fixed at the actual value.

4.3.2 Challenge 1: Multimodality of the Objective Function

Given protein-abundance time-courses $p(t) = (p_1(t), p_2(t), p_3(t))$ and $\widetilde{p}(t) = (\widetilde{p}_1(t), \widetilde{p}_2(t), \widetilde{p}_3(t))$, where $p(t)$ is viewed as the "true" data, we consider the following *relative error*:

$$\sum_{i=1}^{3} \frac{||p_i(t) - \widetilde{p}_i(t)||_2^2}{||p_i(t)||_2^2}. \tag{4.10}$$

Using this relative error as our objective function, we investigate the resulting objective-function landscape. Specifically, the input to the objective function is an alternate set of parameters for system (4.9), and the output is the relative error between the data $p(t)$ from Figure 4.2 and the simulated data $\widetilde{p}(t)$ arising from the alternate parameters and same initial conditions.

Slices of this landscape, displayed in Figure 4.3(a)–(c), show the objective function as a function of only one parameter (all other parameters are fixed at the correct values). This landscape contains many local optima (Figure 4.3), making optimization difficult: gradient-descent methods will get stuck at one of many local optima.

In fact, when we estimated parameters using standard MATLAB optimization commands such as *fminsearch, fmincon,* and *fminbnd* [4], we observed that if an initial guess deviated by more than ten percent of the actual parameter value, then the estimated parameter value differed significantly from the true value. Although in some cases one may be able to guess, for instance, the transcription rate–this rate is closely linked to the maximum protein abundance value (see Section 3.4)–within ten percent, in general having to know the true value within ten percent is a severe limitation. Indeed, one is unlikely to know the Hill coefficient or degradation ratio within such a strict bound.

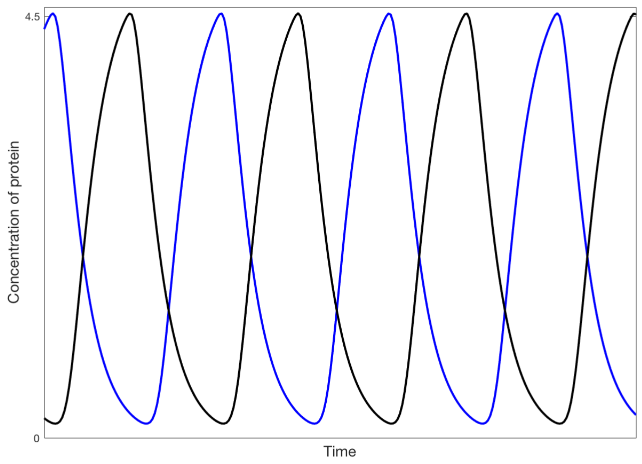


Figure 4.4: Two simulations of the protein abundance $p_3(t)$ with the parameters as in system (4.9) but with differing initial mRNA values. The protein abundances were initialized to be 0 for both simulations. The initial mRNA values for the blue curve were 4, 7, and 3 while the black curve is a simulation with initial mRNA values of 1, 0, and 10.

4.3.3 Challenge 2: Lack of Initial mRNA Values

Traditionally, experimental repressilator data have been generated using fluorescent tags on the three proteins of the system. Thus, data representing protein abundances are collected, but not data for mRNA values. In the context of estimating parameters, not knowing initial mRNA values effectively adds one new parameter for each gene.

As we did for three parameters earlier, we investigate the objective-function landscape with respect to a new parameter, the initial mRNA value for one of the three genes. Displayed in Figure 4.3(d) is the relative error as a function of a single initial mRNA value (fixing all other parameter values and initial mRNA values to their correct values). As we saw for the other parameters, this landscape has many local minima, making the optimization problem challenging.

4.4 Previous Parameter Estimation Procedures Applied to the Repressilator

In 2010, Khammash and Lilacci applied the Extended Kalman Filter [60] to systems biology parameter estimation problems, one in particular being the repressilator [57]. Their new Constrained HEKF Algorithm addressed the two issues of multimodality and lack of initial mRNA values because it was a Bayesian approach that worked independently of any initial values. In Section 4.2.1.1, we outlined the main steps of the algorithm.

Although the Constrained HEKF Algorithm quickly produced relatively accurate estimates of the repressilator compared to other traditional techniques, the identifiability test that Khammash and Lilacci considered, the Variance Test (see Section 4.2.1.2), fails to pass seemingly accurate parameter estimates. To illustrate, in Section 4.4.1, we run the Constrained HEKF Algorithm with the Variance Test on a sample repressilator data set and show that it does not output the best possible parameter estimate.

More recently, Pitt and Banga addressed the issue of multimodality with a novel methodology, GEARS (Global parameter Estimation with Automated Regularization via Sampling) [6]. The method combines three main strategies: (1) global optimization, (2) reduction of search space, and (3) regularized parameter estimation. More information on the derivation and implementation of the algorithm can be found in [6].

To illustrate the effectiveness and speed of the new toolbox, Pitt and Banga applied the new toolbox to four well-known biological oscillators: Goodwin, FitzHugh-Nagumo, Repressilator, and a metabolic oscillator. They showed that the methodology quickly and accurately estimates parameters of the four systems. However, one shortcoming of the GEARS Toolbox is that all initial values of the system are required as input. Below, in Section 4.4.2, we revisit the repressilator example to illustrate how not knowing all initial values can have a significant impact on parameter estimation by the GEARS Toolbox.

4.4.1 Constrained HEKF Algorithm Applied to Simulated Repressilator Data

Below, we apply the Constrained HEKF Algorithm from [57]. We simulate data of system (4.9) with $\alpha_0 = 0$, $\alpha = 5$, $\beta = 3$, and $n = 3$ (red points, Figure 4.5). The data set was taken at a spacing of 1 time unit with a total time of 50. We added normal noise with mean zero and standard deviation .5. Here, we attempt to fit the three parameters α, β, and n.

We ran the Constrained HEKF Algorithm and generated 500 parameter estimates. We then ran the Variance Test with a confidence level of $\gamma = .975$ on each parameter estimate. Only ten parameter estimates passed, listed in Table 4.2. In Table 4.2, the parameter estimates in bold deviate more than ten percent of the original value. Notice from Table 4.2 that, of the three parameters, the estimated Hill coefficient has the most relative variation from its actual value of 3. In fact, seven of the ten Hill coefficients differ from the actual value by more than ten percent (Table 4.2). Our example highlights the need for a better test to discriminate among possible Hill coefficients.

In Figure 4.5, the blue curve plots the model solution of system (4.9) for one of the ten passing parameter estimates (namely, $\alpha = 4.6324$, $\beta = 2.9558$, and $n = 10.2211$). The model solution in blue is very close to the actual model solution in black. However, there are better parameter estimates that did not pass the variance test. For example, each parameter estimate in Table 4.3 has smaller ℓ^1-norm difference between the actual parameter values and the parameter estimate than any of the parameter estimates that passed the Variance Test.

In Section 4.5, we seek a new test that yields more accurate parameter estimates than

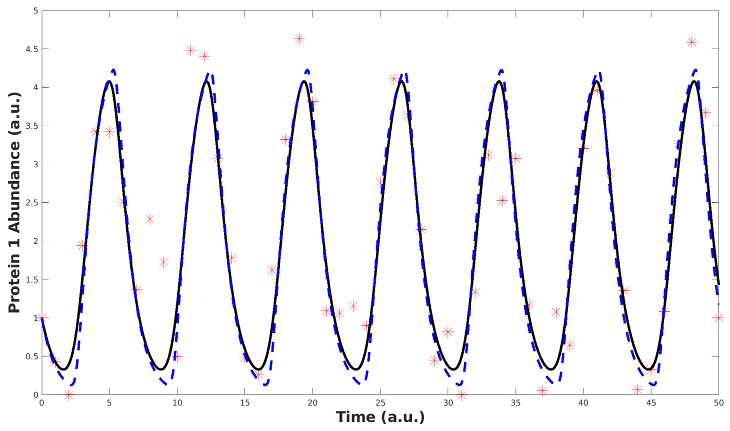


Figure 4.5: The red data points correspond to simulated data of system (4.9) with $\alpha_0 = 0$, $\alpha = 5$, $\beta = 3$, and $n = 3$. The data points were taken from the black curve at a spacing .5 time units, and normal noise with mean zero and standard deviation .5 was added. The blue curve is the model solution of system (4.9) with a parameter set that passed the Variance Test (parameters $\alpha_0 = 0$, $\alpha = 4.6324$, $\beta = 2.9558$, and $n = 10.2211$).

those given by the Variance Test. We consider several possible test statistics that reflect the oscillatory nature of the repressilator such as: period, amplitude, skewness, cost of protein production, etc. In Section 4.5, we investigate how sensitive these quantities are to variations in the three parameters of system (4.9). Using our analysis from Section 4.5, in Section 4.6, we introduce two new algorithms to estimate parameters of the repressilator.

4.4.2 Lack of Some Initial Data Affects Accuracy of Estimates Generated by GEARS

To illustrate the effect of not knowing the initial values on the results from GEARS, we rerun GEARS twice to estimate parameters of system (4.9). To perform the parameter estimation, the GEARS Toolbox requires as input the initial values for all species and time-course abundance levels for at least one species. In the example below, we show that when assuming that not all initial conditions are known, the parameter estimates recovered differ drastically from the true parameter values.

In [6], using system (4.9) with $\alpha = 300$, $\beta = .3$, $n = 8.5$, $\alpha_0 = .05$ and initial conditions $[r_1, r_2, r_3, p_1, p_2, p_3] = [1, .01, 10, 10, 0.01, 1]$, Pitt and Banga simulated data for r_3 and p_3 for

Transcription Rate (true = 5)	Degradation Ratio (true = 3)	Hill Coefficient (true = 3)
4.6324	2.9558	**10.2211**
4.6067	3.0020	**9.9508**
4.6229	2.8655	**9.8716**
4.8377	3.2301	2.8272
4.5490	3.0235	**7.1158**
4.9885	**2.5294**	2.7291
4.4963	3.0080	**8.2367**
4.6844	3.1100	**3.3781**
4.6734	3.1141	**3.4212**
4.2573	2.9487	**5.6124**

Table 4.2: Parameter estimates from data simulated from system (4.9) that passed the Constrained HEKF Algorithm Variance Test. The blue estimate is used to generate the model solution of system (4.9) in Figure 4.5. The estimates in bold deviate more than ten percent of the true parameter value.

Transcription Rate (true = 5)	Degradation Ratio (true = 3)	Hill Coefficient (true = 3)
4.8944	3.0297	3.0092
4.8794	3.0501	2.9482
4.9271	3.0487	2.8880
5.0299	2.9066	2.8527
4.7870	3.0705	3.0785
4.9136	2.9388	2.7427
4.6853	3.1272	2.9879
4.6712	3.0708	3.1262
4.8377	3.2301	2.8272
4.7518	3.0863	3.2388

Table 4.3: The top-ten parameter estimates among the 500 generated as described in Section 4.4.1 when ranked by the ℓ^1-norm difference between the row and the true vector of parameters (5, 3, 3). None of the estimates listed passed the Variance Test.

twenty time points, each spaced ten time units apart. They then used the GEARS Toolbox to fit all four parameters.

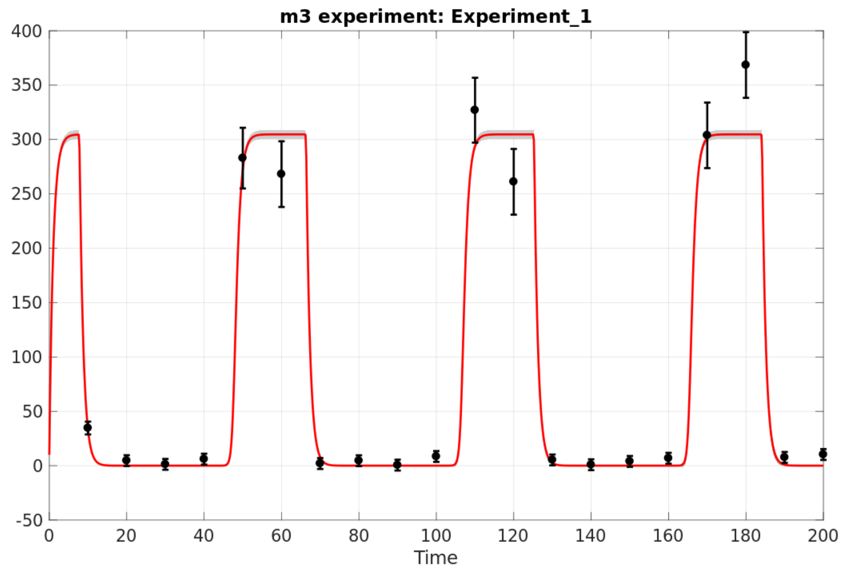
First, we input the correct initial conditions and the simulated data for r_3 and p_3 as done in [6]. For the second run, we input the correct initial conditions for r_1, r_2, r_3, and p_3 and the simulated data sets for r_3 and p_3. However, for p_1 and p_2, we set the initial conditions as $300 \cdot \text{rand}$ where rand is a uniformly distributed rational number between 0 and 1. We choose 300 because the amplitude of the system when $\alpha = 300$ is near 300 (see Section 3.4). Thus, assuming that the system is in the limit cycle, the values p_1 and p_2 should be between 0 and 300 (see Theorem 3.4.1, Section 3.4).

Table 4.4 lists the parameter estimates from the two scenarios. With the correct initial values as input, the parameter estimates of the transcription rate, degradation ratio, and Hill coefficient (row "Original" in Table 4.4) are good approximations of the original parameters. However, after perturbing the initial values for p_1 and p_2, the parameter estimates from GEARS (bottom row of Table 4.4) are drastically different from the actual values. Figure 4.6(a) plots the fit (red curve) of r_3 under the first scenario. Figure 4.6(b) plots the fit of r_3 when we perturb the initial conditions of p_1 and p_2. From Figures 4.6(a) and 4.6(b), we see a drastic difference in the waveforms, periods, and amplitudes between the two fits. Therefore, the GEARS Toolbox fails to produce accurate parameter estimates when some of the initial conditions are unknown.

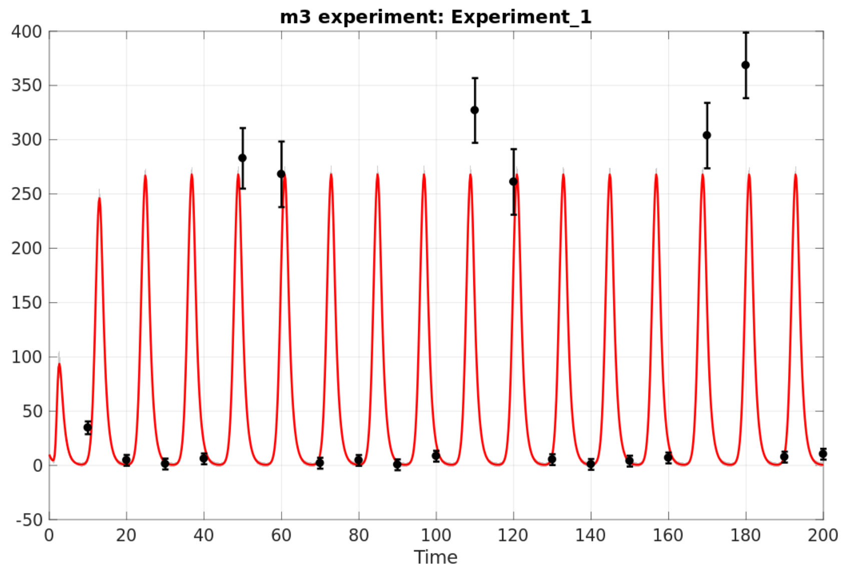
To address the effect of not knowing all initial values, in Section 4.6, we present a novel procedure to extract model solution values at the same time points as the data time points (Algorithm 4.6.3). To circumvent the problem highlighted above in the GEARS example, we couple Algorithm 4.6.3 with an objective function in a global optimization procedure to estimate parameters more accurately for models that exhibit oscillations. Moreover, although we do not do so here, Algorithm 4.6.3 can be integrated into the GEARS Toolbox to improve upon the current version by removing the requirement of knowing all initial values.

4.5 Analyzing Possible Test Statistics

Here, we investigate how certain quantities–e.g., period, amplitude, cost of protein production, skewness, etc.–vary with respect to the parameters of the repressilator system. Ultimately, we seek



(a)



(b)

Figure 4.6: (a) The red curve is the regularized fit of r_3 values of system (4.9) given the estimate from GEARS with input as the correct initial values and r_3 and p_3 time-course data (r_3 data points in black). (b) The red curve is the regularized fit of r_3 values of system (4.9) given the estimate from GEARS with input as the perturbed initial values of p_1 and p_2 and all other inputs the same as in the original example.

Run	Leakiness α_0 (true = 0.05)	Transcription Rate (true = 300)	Degradation Ratio (true = .3)	Hill Coefficient (true = 8.5)
Original	.001	304.7459	.2969337	8.835106
Perturbed	0.001676976	500	2.933361	1.954551

Table 4.4: Parameter estimates generated by the GEARS Toolbox. The original row lists parameter estimates that were found from GEARS with input as the r_3 and p_3 data sets and correct initial values. The perturbed row lists the parameter estimates that were found from GEARS with input as the perturbed initial values for p_1 and p_2 and all other input the same as in the original example. See Section 4.4.2 for how the initial values were perturbed.

a test statistic that includes one or more such quantities as a means to accept or reject parameter estimates generated by the Constrained HEKF Algorithm. Recall that, in Section 4.4.1, we saw that the variance test rejected parameter estimates that were more accurate than those it passed, when considering the ℓ^1-norm between the parameter estimate and the vector of actual parameter values.

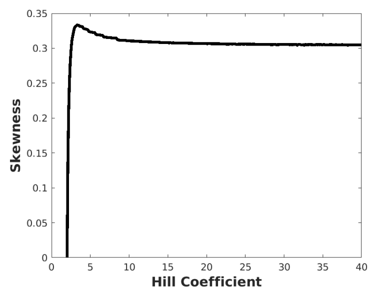
4.5.1 Methods

To determine how sensitive the various quantities are to the parameters, using system (4.9) with $\alpha_0 = 0$, we investigate three scenarios. For each test statistic, we examine how it varies with respect to the Hill coefficient n when fixing $\alpha = 5$ and $\beta = 3$. Similarly, we investigate how the test quantity varies with respect to α when fixing $n = 3$ and $\beta = 3$. Finally, we consider how the test statistic varies with respect to β when fixing $\alpha = 5$ and $n = 3$.

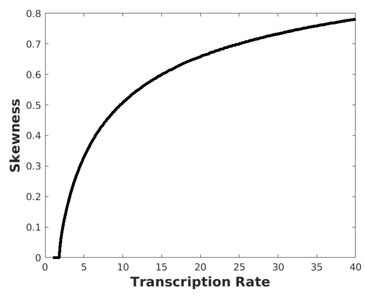
4.5.2 Skewness

First, we investigate how the skewness of a data set varies with respect to the three parameters in system (4.9). To compute the skewness, we simulate system (4.9) to the limit cycle. Once in the limit cycle, we extract protein 1 abundance levels at a uniform spacing of .01 for one period. We then use Eqn. (4.3) to estimate the skewness.

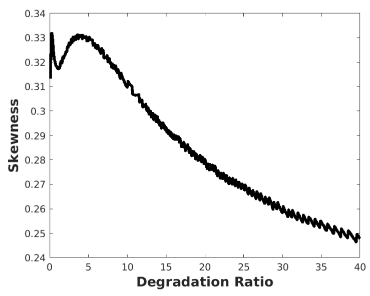
Figure 4.7(a) plots how the skewness varies with respect to the Hill coefficient. The skewness stabilizes to a value near 0.3 near a Hill coefficient of 5 (Figure 4.7(a)). Figure 4.7(b) plots the skewness with respect to the transcription rate. The skewness increases monotonically and appears



(a)



(b)



(c)

Figure 4.7: Plot of how the skewness varies with respect to (a) the Hill coefficient, (b) the transcription rate, and (c) the degradation ratio. The parameters were fixed at $\alpha = 5$, $\beta = 3$, and $n = 3$ when not considered as a variable. **Code used to obtain plots is found in Appendix A, Section A.2.2**.

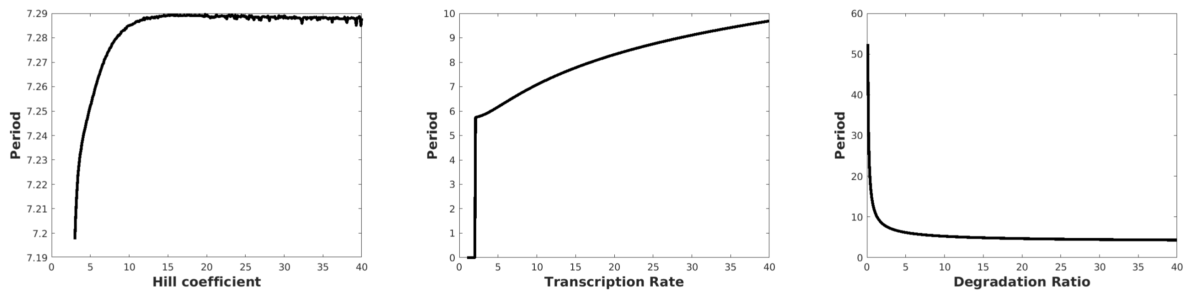
to approaches a skewness value of 1 (Figure 4.7(b)). Recall, from Section 4.2.2, that a skewness value near 1 indicates a tail to the right (Figure 4.1(c)). This result corroborates our intuition from numerical simulations that, with increasing transcription rate, the waveform skews with a longer tail to the right.

Finally, Figure 4.7(c) plots the variation in skewness with respect to the degradation ratio, β. Initially, the skewness oscillates around a value of 0.32. Then, after a degradation ratio of about 5, the skewness decreases with increasing degradation ratio.

4.5.3 Period

Second, we consider how the period varies with respect to the three parameters (Figure 4.8). Figure 4.8(a) plots how the period varies with respect to the Hill coefficient. When fixing $\alpha = 5$ and $\beta = 3$, the Hopf bifurcation with respect to n occurs near $n = 2.6$, so we begin the plot in Figure 4.8(a) at that Hill coefficient. From Figure 4.8(a), we see that the period does not vary more than approximately 1.4% and stabilizes near a Hill coefficient of 10.

Figure 4.8(b) plots the variation in the period with respect to the transcription rate. When fixing $\beta = n = 3$, the Hop bifurcation with respect to α happens near $\alpha = 1.9$ (Figure 4.8(b)). The period appears more sensitive to the transcription rate than to the Hill coefficient and continues to increase with increasing transcription rate (Figure 4.8(b)). However, from Figure 4.8(c), we see



(a) (b) (c)

Figure 4.8: Plot of how the period varies with respect to (a) the Hill coefficient, (b) the transcription rate, and (c) the degradation ratio. The parameters were fixed at $\alpha = 5$, $\beta = 3$, and $n = 3$ when not considered as a variable. **Code used to obtain plots is found in Appendix A, Section A.2.3**.

that, among the three parameters, the period is most sensitive to the degradation ratio. In fact, when fixing $\alpha = 5$ and $n = 3$, at the Hopf bifurcation near $\beta = .1$, the period is above 50 and decreases to less than ten when $\beta = 40$.

4.5.4 Amplitude

Next, we investigate how the amplitude varies with respect to the three parameters. By amplitude, we mean the maximum protein-1 abundance. We choose this characterization because it is easy to calculate from a discrete data set with noise. Figure 4.9(a) plots the amplitude with respect to the Hill coefficient when fixing $\alpha = 5$ and $\beta = 3$. Similar to Figure 4.8(a), the amplitude stabilizes near a Hill coefficient of 7 (Figure 4.9(a)).

From Figure 4.9(b), we see that the amplitude satisfies a near-linear relationship with respect to the transcription rate. Recall, from Section 3.4, that we showed that the upper bound of the protein abundance level of system (4.9) is linear with respect to α. Also, we showed that, for system (4.9), the protein 1 abundance level closely follows the upper bound. Therefore, it is not surprising that we see a near linear relationship between the amplitude and the transcription rate in Figure 4.9(b).

Lastly, in Figure 4.9(c), we plot the variation of the amplitude with respect to the degradation ratio. From Figure 4.9(c), we see that, initially, the amplitude increases until approximately $\beta = 1.9$ and then decreases.

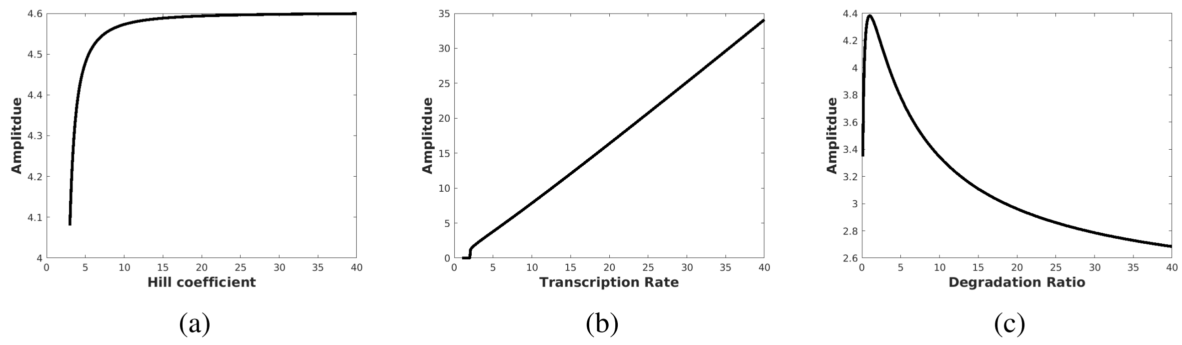


Figure 4.9: Plot of how the amplitude varies with respect to (a) the Hill coefficient, (b) the transcription rate, and (c) the degradation ratio. The parameters were fixed at $\alpha = 5$, $\beta = 3$, and $n = 3$ when not considered as a variable. **Code used to obtain plots is found in Appendix A, Section A.2.3**.

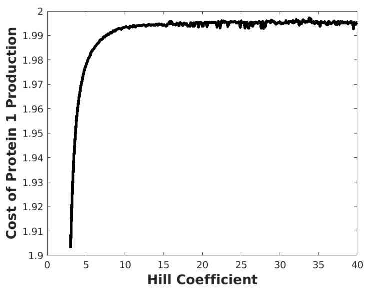
4.5.5 Cost of Protein Production

In Section 4.2.3, we recall the definition of cost of protein production introduced by Jo *et al.* in 2018 [59]. We consider the cost as a test quantity for the following reasons: the cost is easy to estimate from discrete data, the cost is tied to the shape of the protein waveforms, and the cost has implications in the evolution of biological clocks [59].

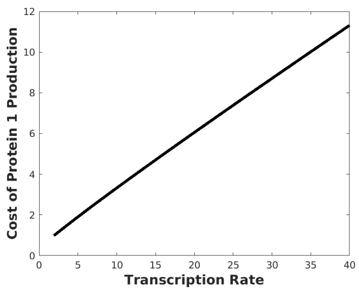
Here, we investigate how sensitive the cost quantity is to the three parameters. Figure 4.10(a) plots the variation in cost with respect to the Hill coefficient. Again, as in Figures 4.8(a) and 4.9(a), the cost stabilizes near a Hill coefficient of 10 with a total variation of about 5%. The other two plots, Figures 4.10(b) and 4.10(c), are nearly identical in shape to amplitude figures (Figures 4.9(b) and 4.9(c)). The cost increases nearly linearly with respect to the transcription rate (Figure 4.10(b)). In contrast, the cost initially increases with respect to the degradation ratio (from $\beta \approx .1$ to $\beta \approx 2$, Figure 4.10(c)) and then decreases.

4.5.6 Peak-to-Trough Time

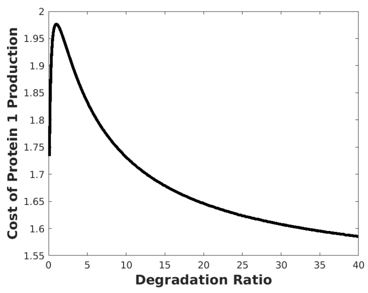
The quantities considered, thus far, have shown variation with respect to the transcription rate and degradation ratio. However, the quantities stabilized quickly with respect to the Hill coefficient. Consequently, for the remainder of the section, we investigate more quantities in an effort



(a)



(b)



(c)

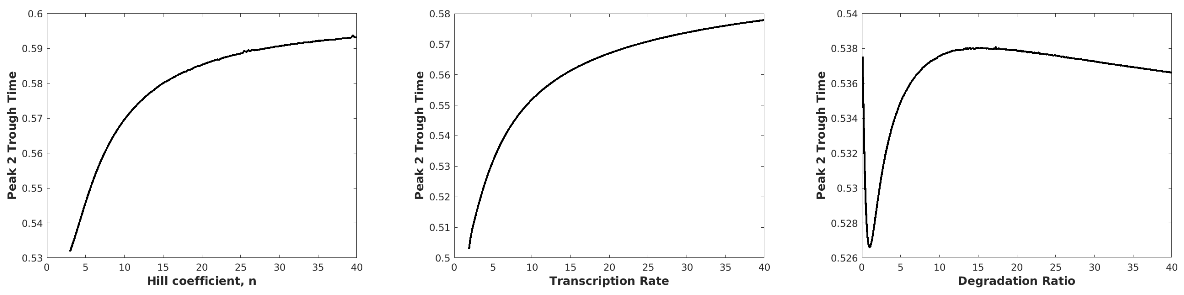
Figure 4.10: Plot of how the cost varies with respect to (a) the Hill coefficient, (b) the transcription rate, and (c) the degradation ratio. The parameters were fixed at $\alpha = 5$, $\beta = 3$, and $n = 3$ when not considered as a variable. **Code used to obtain plots appears in Appendix A, Section A.2.4**.

to identify one sensitive to the Hill coefficient. Recall, from our example in Section 4.4.1, that the Hill coefficient, of the three parameters, expressed the highest relative variation from the true values among the estimates accepted by the Variance Test.

First, we investigate how the time from the peak to the trough varies with respect to the three parameters. We saw from numerical simulations of system (4.9) with varying Hill coefficients that, as the Hill coefficient increases, the steepness of the degradation portion of the protein abundance waveform increases. Therefore, we expect that the time from peak to trough will decrease with respect to the Hill coefficient.

Figure 4.11(a) plots the variation in the time from peak to trough with respect to the Hill coefficient. We normalize the time by the period of system (4.9). From Figure 4.11(a), we see that the time increases with increasing Hill coefficient, which is the opposite of our original hypothesis mentioned above. Apparently, even though the degradation portion of the protein waveform is sharper with a larger Hill coefficient, the time that the protein abundance spends near the trough is greater with respect to the period.

Figure 4.11(b) plots the variation in the time between the peak and trough with respect to the transcription rate. Interestingly, the plot follows similarly to the plot in Figure 4.11(a). With respect to the degradation rate, however, the peak to trough time initially decreases before increasing. Perhaps, since we are normalizing by the period of the system, the initial decrease corresponds to



(a) (b) (c)

Figure 4.11: Plot of how the time between the peak and the trough varies with respect to (a) the Hill coefficient, (b) the transcription rate, and (c) the degradation ratio. The parameters were fixed at $\alpha = 5$, $\beta = 3$, and $n = 3$ when not considered as a variable. **Code used to obtain plots appears in Appendix A, Section A.2.5**.

the initial increase in the period seen in Figure 4.8(c).

4.5.7 Peak to 50%-Amplitude

In light of Figure 4.11(a), we consider a variant of the peak to trough time. Instead of the time taken from the peak to trough, we now investigate how the time taken from the peak to 50% of the amplitude varies with respect to the three parameters. Figure 4.12 plots an example of the quantity. The red line corresponds to the time when the protein level is 50% of the amplitude. Then the test quantity is the time at the red line. As in Section 4.5.6, we normalize the time by the period.

Since the increase in Figure 4.11(a) was due to the time that the protein abundance spends near the trough, we expect that considering the reduced time from the peak to 50% of the amplitude will reveal more variation with respect to the three parameters. In fact, Figure 4.13(a) plots the variation in this time with respect to the Hill coefficient. As expected, the time decreases with respect to increasing Hill coefficient. However, similar to the other quantities considered above, the time stabilizes near a Hill coefficient value of 10.

Figures 4.13(b) and 4.13(c) plot the quantity with respect to the transcription rate and degradation ratio, respectively. As in the analysis in Section 4.5.6, the variation with respect to the transcription rate matches the plot with respect to the Hill coefficient. As in Section 4.5.6, the test quantity is influenced heavily by the normalization by the period resulting in an initial decrease

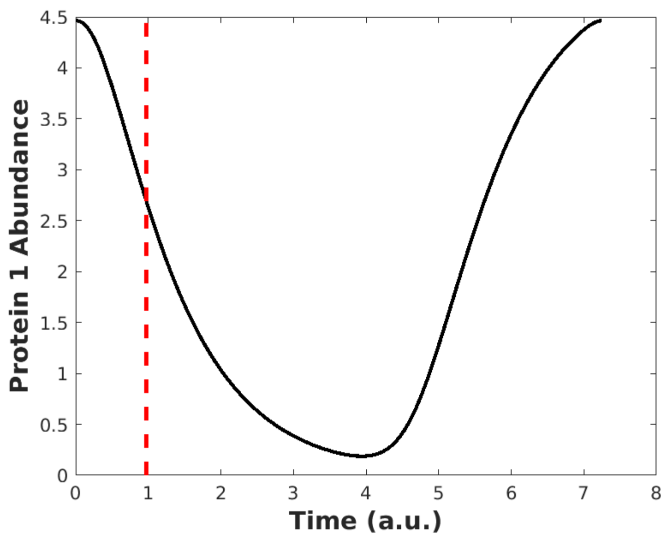


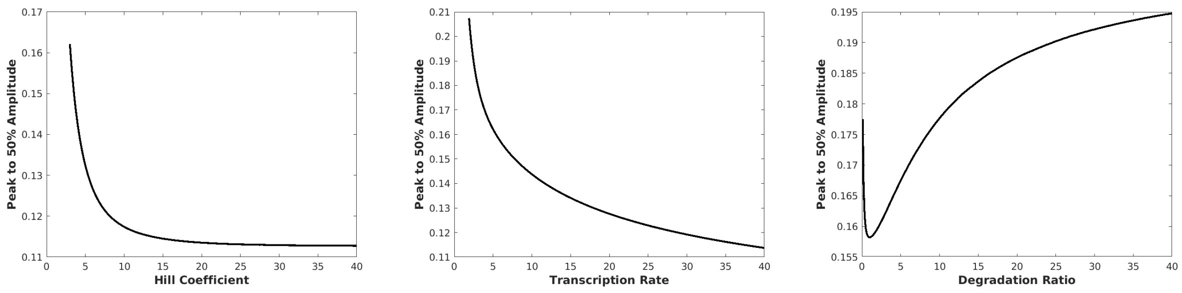
Figure 4.12: Example of the test quantity considered in Section 4.5.7. One period of protein abundance levels of system (4.9) with $\alpha = 5$, $\beta = 3$, and $n = 3$. The red line corresponds to the time when the protein abundance level is half of the amplitude.

and then an increase in the time.

4.5.8 Minimum Derivative

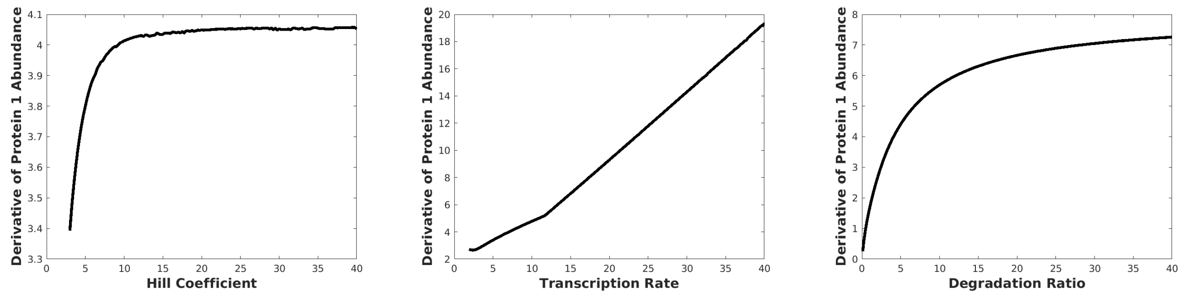
Finally, we consider one more test statistic: the magnitude of the minimum derivative of the protein abundance level. Similar to the quantities in Sections 4.5.6 and 4.5.7, we consider this value because of the increased sharpness in the degradation portion of the protein abundance waveform. We expect that, with increasing Hill coefficient, the magnitude of the minimum derivative will increase. In fact, Figure 4.14(a) shows that the minimum derivative in magnitude does increase with respect to the Hill coefficient. However, as with all the other test quantities considered, it stabilizes near a Hill coefficient of 10.

Figure 4.14(b) plots the magnitude of the minimum derivative with respect to the transcription rate. The derivative appears to increase linearly with respect to the transcription rate (Figure 4.14(b)). Similarly, the derivative initially increases linearly with respect to the degradation ratio, but it levels off instead of increasing linearly (Figure 4.14(c)).



(a) (b) (c)

Figure 4.13: Plot of how the time from the peak to 50% of the max protein 1 level varies with respect to (a) the Hill coefficient, (b) the transcription rate, and (c) the degradation ratio. The parameters were fixed at $\alpha = 5$, $\beta = 3$, and $n = 3$ when not considered as a variable. **Code used to obtain plots is found in Appendix A, Section A.2.6**.



(a) (b) (c)

Figure 4.14: Plot of how the minimum derivative (in magnitude) varies with respect to (a) the Hill coefficient, (b) the transcription rate, and (c) the degradation ratio. The parameters were fixed at $\alpha = 5$, $\beta = 3$, and $n = 3$ when not considered as a variable. **Code used to obtain plots appears in Appendix A, Section A.2.7**.

4.5.9 Biological Significance

In this section, we investigate how sensitive various quantities of systems with oscillating components are to the Hill coefficient, transcription rate, and degradation ratio. We predominantly saw that these values were only sensitive to smaller Hill coefficients ($n \approx$ 3-5) and stabilized soon after ($n \approx$ 5-10). In light of Remark 3.1.1, there is a significant biological interpretation of this discovery: *For biological clocks generated by transcriptional regulation, increasing the number of transcription factors has a negligible impact on the shape and period of the oscillations.* This phenomenon may help to explain why, in general, many transcription factors function with at most two or three others on a specific promoter. Thus, it is not surprising that, in biology, it is rare to see more than three proteins bind to a gene promoter.

4.6 New Algorithms

In Section 4.5, we worked towards a quantity, or quantities, that we can incorporate into an analogue of the Variance Test that will yield more accurate parameter estimates. We found that, after analyzing the variation in the quantities with respect to each parameter, most quantities were sensitive to the transcription rate and the degradation ratio, but not the Hill coefficient, so we choose as test statistics the quantities that are most easily estimated from a discrete data set: period, amplitude, and cost of protein production. Recall that, in Section 4.2.4, we review how to estimate the period given a discrete data set of protein values.

Here, we incorporate the test statistics period, amplitude, and cost of protein production into an identifiability test coupled with the Constrained HEKF Algorithm. In particular, we introduce a new algorithm to estimate parameters of the repressilator (system (4.9)). The algorithm initially generates many possible parameter estimates using the Constrained HEKF Algorithm (Algorithm 4.2.1) initialized in a certain way. Then, the algorithm, outlined below, ranks the parameter estimates by the absolute errors between amplitude, cost, and period in the model solution with the estimated parameters and those from the data set.

Algorithm 4.6.1. Enhanced Constrained HEKF Algorithm #1:

INPUT:	Time points: $t_1, \dots, t_k$. Protein data points: $p_1^{(i)}, \dots, p_k^{(i)}$ for $i =$ 1, 2, and 3. Model (4.9). Weights: $\lambda_{\mathcal{C}}$, $\lambda_{\mathcal{T}}$, and $\lambda_{\mathcal{A}}$.
OUTPUT:	Parameter estimate $\hat{\theta}$.

1. Estimate the period ($\widetilde{\mathcal{T}}$), cost ($\widetilde{\mathcal{C}}$), and amplitude ($\widetilde{\mathcal{A}}$) of the data set.
   (a) In Appendix A, Section A.2.10, we give the code used to estimate the cost.
2. Initialize Algorithm 4.2.1.
   (a) For system (4.9), in particular, initialize the transcription rate α as the maximum protein abundance.
   (b) We initialize the matrix Q from Algorithm 4.2.1 as $5 \cdot$ rand $\cdot\, I_{2m}$ where m is the number of genes in system (4.9), rand is a uniformly distributed random rational number between 0 and 1, and I_{2m} is the $2m \times 2m$ identity matrix.
   (c) We initialize the state $r_i(0)$ as a uniformly distributed random rational number from 0 to $\max\{p_1^{(i)}, \dots, p_k^{(i)}\}$.
3. Run Algorithm 4.2.1 and save the parameter estimates.
4. Repeat steps 2 and 3 for as many times as desired. **For our runs of this algorithm, we generate 600 parameter estimates.**
5. For each parameter estimate generated, estimate the cost ($\mathcal{C}^*$), period ($\mathcal{T}^*$), and amplitude ($\mathcal{A}^*$) of the system in the limit cycle.
   (a) Compute the cost at the model values found using the Event Location procedure outlined in Algorithm 4.6.3.

(b) Use the Event Location procedure outlined in Algorithm 4.6.3 to compute an estimate of the period.

6. Compute the absolute error:

$$\epsilon = \lambda_{\mathcal{C}}|\mathcal{C}^* - \widetilde{\mathcal{C}}| + \lambda_{\mathcal{T}}|\mathcal{T}^* - \widetilde{\mathcal{T}}| + \lambda_{\mathcal{A}}|\mathcal{A}^* - \widetilde{\mathcal{A}}|, \tag{4.11}$$

for weights $0 < \lambda_{\mathcal{C}}, \lambda_{\mathcal{T}}, \lambda_{\mathcal{A}}$. **For our run of the algorithm in Section 4.7.1, we use equal weights.**

7. Rank the parameter estimates from smallest to largest with respect to ϵ.

8. **Output** $\hat{\theta}$ with the smallest ϵ.

Next, while Algorithm 4.6.1 above considers amplitude, cost, and period as test statistics, we introduce a second algorithm that ranks the parameter estimates by the ℓ^2-norm error between the protein values of the data set versus those of the model solution with the parameter estimate. The second algorithm is outlined below.

Algorithm 4.6.2. Enhanced Constrained HEKF Algorithm #2:

INPUT:	Time points: $t_1, \ldots, t_k$. Protein data points: $p_1^{(i)}, \ldots, p_k^{(i)}$ for $i = 1$, 2, and 3. Model (4.9).
OUTPUT:	Parameter estimate $\hat{\theta}$.

1. Initialize Algorithm 4.2.1.

(a) For system (4.9), in particular, initialize the transcription rate α as the maximum protein abundance.

(b) Initialize the matrix Q from Algorithm 4.2.1 as $5 \cdot \text{rand} \cdot I_{2m}$ where m is the number of genes in system (4.9), rand is a uniformly distributed random number between 0 and 1, and I_{2m} is the $2m \times 2m$ identity matrix.

(c) Initialize the state $r_i(0)$ as a uniformly random number from 0 to $\max\{p_1^{(i)}, \ldots, p_k^{(i)}\}$.

2. Run Algorithm 4.2.1 (Section 4.2.1.1) and save the parameter estimates.

3. Repeat Steps 2 and 3 for as many times as desired. **We suggest at least 100 repetitions.**

4. For each parameter estimate generated, $\hat{\theta}$, run Algorithm 4.6.3 to estimate model values $\hat{p}_1^{(i)}, \ldots, \hat{p}_k^{(i)}$ at the data time points $t_1, \ldots, t_k$ for $i = 1$, 2, and 3.

5. For all three proteins, $i = 1, 2, 3$, compute the relative error, ϵ_i, between the model solution values found in Step 4 and the data values:

$$\epsilon_i = \frac{||\hat{p}^{(i)} - p^{(i)}||}{||p^{(i)}||}.$$

6. **Output** $\hat{\theta}$ with the smallest $\sum_{i=1}^{m} \epsilon_i$ where m is the number of genes in system (4.9).

The event location procedure used in the Algorithms 4.6.1 and 4.6.2 is given below.

Algorithm 4.6.3. The **Event Location Procedure** to compute model estimates at the same time points as the data:

INPUT:	Parameter estimate $\hat{\theta}$ for system (4.9). Time points $t_1, \ldots, t_k$. Protein data points: $p_1^{(1)}$ and $p_2^{(1)}$.
OUTPUT:	**REJECT** or: 1. Period of system (4.9) with parameter estimate $\hat{\theta}$. 2. The model solution values $\bar{p}_1^{(i)}, \ldots, \bar{p}_k^{(i)}$ for $i = 1$, 2, and 3.

1. Run system (4.9) with the parameter estimate $\hat{\theta}$ to a limit cycle or to a steady state (Recall, from Section 2.3.4, that these dynamics are the only two possibilities for the 3-gene repressilator).

   (a) If the model converges to a steady state, **REJECT** the parameter estimate and **END**.

2. With the model solution in the limit cycle, save two consecutive time points, denoted $\hat{t}_1$ and $\hat{t}_2$, when the model satisfies the two conditions: (1) the protein-1 value of the model with $\hat{\theta}$ matches $p_1^{(1)}$ and (2) the direction of protein-1–a value of 1 if protein-1 is increasing and a value of -1 if protein-1 is decreasing–matches $\text{sign}(p_2^* - p_1^*)$.

   (a) If the model never satisfies both conditions while in the limit cycle, **REJECT** the parameter estimate and **END**.

   (b) Let $(\hat{r}_1, \ldots, \hat{r}_n, \hat{p}_1, \ldots, \hat{p}_n)$ and $\hat{t}$ denote the abundance values and time, respectively, when the model with $\hat{\theta}$ satisfies the two conditions.

   (c) **Output** the period estimated by the time $\hat{t}_2 - \hat{t}_1$.

3. Run system (4.9) with parameter estimate $\hat{\theta}$ using as initial conditions $(\hat{r}_1, \ldots, \hat{r}_n, \hat{p}_1, \ldots, \hat{p}_n)$ and a time span given by the time points of the data, $t_1, \ldots, t_k$.

   (a) When using an ODE solver such as *ode23tb* in MATLAB, if one specifies a time span of discrete points, then the output is model solutions at those time points specifically.

   (b) **Output** the protein-i values $\bar{p}_1^{(i)}, \ldots, \bar{p}_k^{(i)}$ for $i = 1$, 2, and 3.

Remark 4.6.4. For the above Algorithm 4.6.3, we implement Step 2 in MATLAB using the "Events" options in the *odeset* function. The "Events" option takes an events function that outputs three values: (1) a value, (2) whether finding the event stops the ode solver (isTerminal: 0 if no, 1 if yes), and (3) the direction of the value.

(i) Set the value as $p_1(t) - p_1^*$.

(ii) Set isTerminal = 0.

(iii) Set the direction as 1 if the initial direction of p_1^* is increasing and -1 if the initial direction of p_1^* is decreasing.

Remark 4.6.5. The Event Location Procedure outlined above in Algorithm 4.6.3 addresses the challenge discussed in Section 4.3.3, the lack of initial mRNA values. By matching the protein-1 level of the model solution with the parameter estimate, the protein-1 values are initialized in the same phase as the protein-1 values in the data set. Thus, the relative error computed in Step 5 in Algorithm 4.6.2 is due to differences in the waveform of the model solution with the parameter estimate versus the data, not due to the model solution values being in a different phase from the data points.

Remark 4.6.6. We recommend combining Algorithm 4.6.3 with a traditional objective function to circumvent the challenge of the lack of initial mRNA values. To illustrate this, in Section 4.8.1, we employ Algorithm 4.6.3 with a particle swarm optimization procedure, and then compare the results to our final algorithm outlined in Section 4.8.

4.7 Algorithm Results

First, we run Algorithms 4.6.1 and 4.6.2 on simulated data sets generated from system (4.9) with $\alpha = 5$, $\beta = 3$, $n = 3$, and $\alpha_0 = 0$. The black curve in Figure 4.15(a) plots the model solution of system (4.9) for a total time of 50 (a.u.). We use both algorithms to estimate the parameters α, β, and n.

Remark 4.7.1. Ordinarily, circadian rhythm experiments generate data every two to four hours resulting in about six to twelve time points per period. The period of protein abundances in system (4.9) is around 7.2. Thus, if we simulate data with a spacing of .5 or 1, we generate data sets that have around seven to fourteen data points per period, reflecting the nature of experimental data.

In all, we generate eight data sets by simulating data sets with a uniform spacing of .5 or 1 time unit (a.u.) and with Gaussian random noise added at each time point with standard deviations 0, .1, .25, and .5. Henceforth, we label a data set (s, σ) where $s = 0.5$ or 1 is the spacing and $\sigma = 0, .1, .25,$ or .5 is the standard deviation of the added noise. **We give the code used to generate the data sets in Appendix A, Section A.2.9**.

In Appendix B, Section B.2.1, we plot all the data sets that we generated. In Figure 4.15, we plot the data sets (0.5, 0) and (1, 0.5).

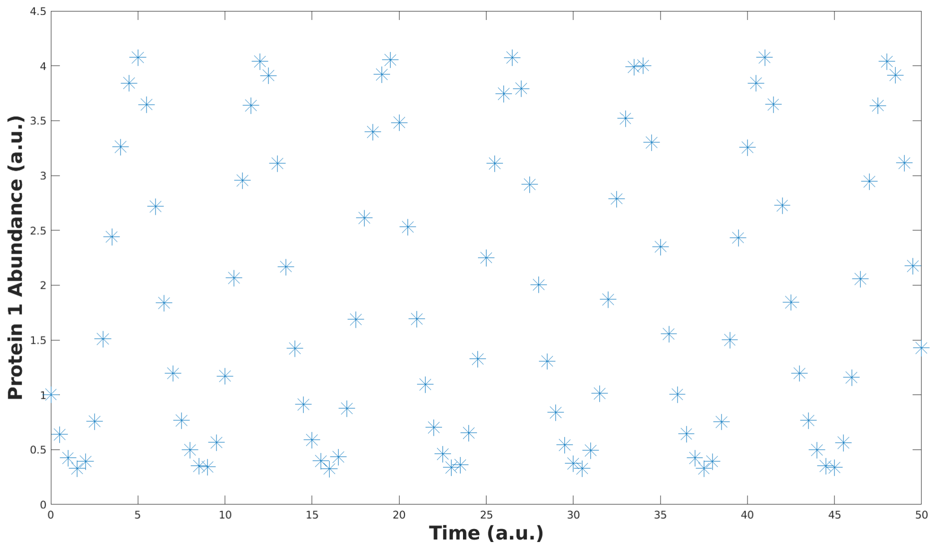
4.7.1 Algorithm 4.6.1 Results

We ran Algorithm 4.6.1 on all eight data sets. We chose equal weights for Step 6 in Algorithm 4.6.1. Table 4.5 lists the parameter estimates generated by Algorithm 4.6.1 for the eight simulated data sets. In general, the recovered parameter estimates are exceptionally accurate. With the exception of one Hill coefficient (3.488) estimated using the data set (0.5, 0.5) (Table 4.5), no parameter estimate deviated by more than ten percent of the original value. In Appendix B, Section B.1.1, we include tables that list the top-ten estimates from Algorithm 4.6.1 for each of the data sets and each of the test statistics.

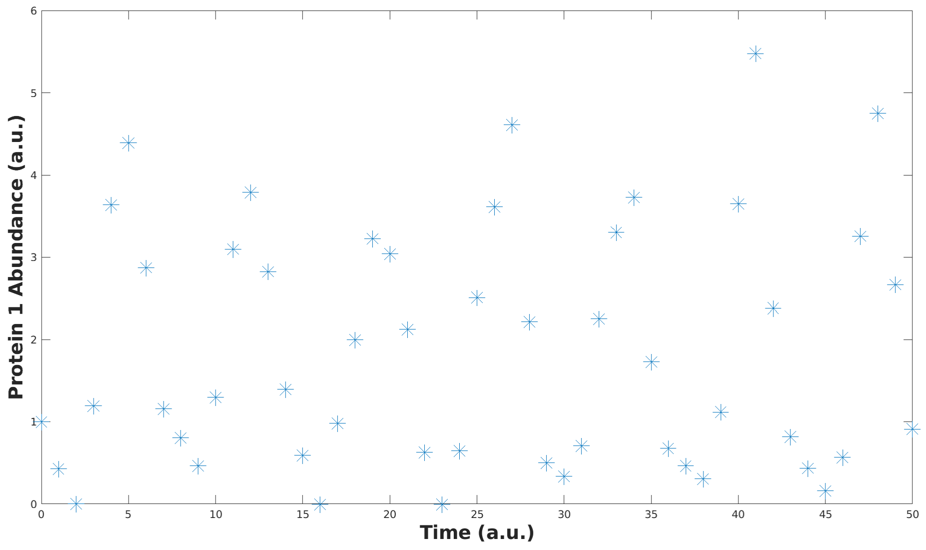
In addition to ranking with respect to the absolute error in Step 6 in Algorithm 4.6.1, we also rank the parameter estimates based on individual test statistics: amplitude, cost, and period. Table 4.6 lists the parameter estimates generated by Algorithm 4.6.1 when ranking with amplitude only (Column 3), cost only (Column 4), and period only (Column 5). In general, the parameter estimate given by the cost test statistic is the most accurate among the three test statistics. In Table 4.6, we color the cost parameter estimate blue if it is more accurate than both of the corresponding estimates in Tables 4.5 and 4.7.

Next, we plot the model solutions given the top parameter estimate for each test statistic (Figure 4.17) and the final parameter estimate generated by Algorithm 4.6.1 (Figure 4.16). The data set in Figures 4.16 and 4.17 is the data set (1, 0.5). The black curve in both figures is the model solution of system (4.9) with the true parameters. Figure 4.16 plots the model solution of system (4.9) with parameters given by the top ranked estimate in the last row in Table 4.5. Similarly, Figure 4.17 plots the model solutions of system (4.9) with parameters given by the estimates in the last row of Table 4.6 (the top ranked estimates for amplitude, cost, and period, respectively).

From Figure 4.16, we see that the model solution with parameter estimate output from Algorithm 4.6.1 is nearly identical to the model solution with the true parameter values. Likewise, from Figure 4.17(b), we see that the model solution of system (4.9) with the parameter estimate given



(a)



(b)

Figure 4.15: (a) Simulated data set (0.5, 0) from system (4.9). (b) Simulated data set (1, 0.5) from system (4.9). After adding noise, if any data point is negative, we set that data point to 0. System (4.9) was simulated to the limit cycle using initial conditions. **Code used to generate these two data sets and all the others appears in Appendix A, Section A.2.9.**

by the cost test statistic also aligns well with the actual model solution.

Finally, we investigate the variation of parameters ranked in the top 25 estimates given by each test statistic and the absolute error in Step 6 in Algorithm 4.6.1. In Figure 4.18, we plot the frequency of estimated transcription rates that were generated by the 600 runs of Step 4 in Algorithm 4.6.1 (white histograms). For each test statistic, we then plot the frequency of the top 25 transcription rates (red histogram). The first row of Figure 4.18 corresponds to the weighted sum, the second row the amplitude statistic, the third row the cost statistic, and the last row the period statistic. Notice from Figure 4.18 that the amplitude gives the smallest variation in the top 25 transcription rates. This is not surprising, as the amplitude is closely related to the transcription rate (recall discussion in Section 3.4).

Along with the transcription rate, in Figures 4.19 and 4.20, we plot analogous histograms for the degradation ratio and Hill coefficient, respectively. From Figure 4.19, we see that the top degradation ratio estimates are situated near the actual value of 3 (with the exception of the cost statistic which gives a top 25 estimate near 3.8). In contrast, the estimates of the Hill coefficient vary significantly from the actual value of 3 (Figure 4.20). With the exception of the cost test statistic, top 25 estimates of the Hill coefficient are seen near 10 (Figure 4.20). This follows from our analysis in Section 4.5 in which we saw a lack of sensitivity of variation in the Hill coefficient to the various test statistics considered.

4.7.2 Algorithm 4.6.2 Results

We ran Algorithm 4.6.2 on the same eight data sets, two of which are given in Figure 4.15. Table 4.7 lists the top parameter estimate for each data set. In Table 4.7, the estimates colored red correspond to estimates that are less accurate than the corresponding estimate generated by Algorithm 4.6.1 listed in Table 4.5. In general, the parameter estimates generated by Algorithm 4.6.2 are more accurate than those generated by Algorithm 4.6.1, which, from Section 4.7.1, are already very accurate. From the last column in Tables 4.5 and 4.7, we see that, because a majority of the time for both algorithms is in generating all the parameter estimates, the elapsed time is nearly identical.

Data Set (s, σ)	TR (true = 5)	DR (true = 3)	Hill (true = 3)	Elapsed Time (minutes)
(0.5, 0)	5.01025578	2.99585437	2.97951441	22.65
(1, 0)	5.00321990	2.99992929	2.99337307	20.31
(0.5, 0.1)	4.98654296	2.98740773	2.98848187	19.83
(1, 0.1)	5.15732470	2.99143751	2.81515213	20.86
(.5, 0.25)	5.18151647	3.00361258	2.80073595	18.79
(1, .0.25)	4.90739488	2.88710651	3.19750507	23.96
(0.5, 0.5)	4.76019839	2.96649153	3.48843037	24.85
(1, 0.5)	4.89380202	2.94513595	3.09510014	19.50

Table 4.5: Parameter estimates generated by Algorithm 4.6.1 for the respective time spacing and noise added for the data in Appendix B. We repeat Step 4 600 times. Actual parameter values given in system (4.9) are 5, 3, and 3 for the transcription rate (TR), degradation ratio (DR), and Hill coefficient (Hill), respectively.

Data Set	Amplitude-Ranking Estimate			Cost-Ranking Estimate			Period-Ranking Estimate		
(0.5, 0)	5.0366	2.9987	2.9467	**4.9990**	2.9926	**3.0000**	5.0102	2.9958	2.9795
(1, 0)	5.0023	2.9614	2.9797	**5.0009**	2.9980	**2.9993**	5.0029	3.0009	2.9956
(0.5, 0.1)	4.6523	2.6873	3.7524	**4.9865**	2.9874	**2.9884**	5.0373	2.9955	2.9121
(1, 0.1)	4.5140	2.8629	5.7750	**5.0329**	3.0129	**2.9935**	4.5140	2.8629	5.7750
(0.5, 0.25)	5.0907	2.9678	2.8777	5.2399	3.0299	2.7850	5.1831	3.0131	2.8160
(1, 0.25)	4.4685	2.9544	9.8917	**4.9481**	2.8524	3.2983	4.5969	2.8940	6.3650
(0.5, 0.5)	4.7601	2.966	3.4884	**4.8924**	**3.0331**	**3.0342**	4.7601	2.966	3.4884
(1, 0.5)	4.5994	2.8729	4.0605	4.9227	**2.9741**	**2.9775**	3.9820	2.7443	3.7897

Table 4.6: Parameter estimates generated by Algorithm 4.6.1 when ranked by the amplitude, cost, and period. We repeat Step 4 600 times. Within the columns for the specific test statistic, the parameters are ordered: transcription rate (TR), degradation ratio (DR), and Hill coefficient (Hill). Actual parameter values given in system (4.9) are 5, 3, and 3 for the TR, DR, and Hill, respectively. In the Cost Estimate column, the values that are blue/bold are parameter estimates that are better than the corresponding estimates in Tables 4.5 and 4.7.

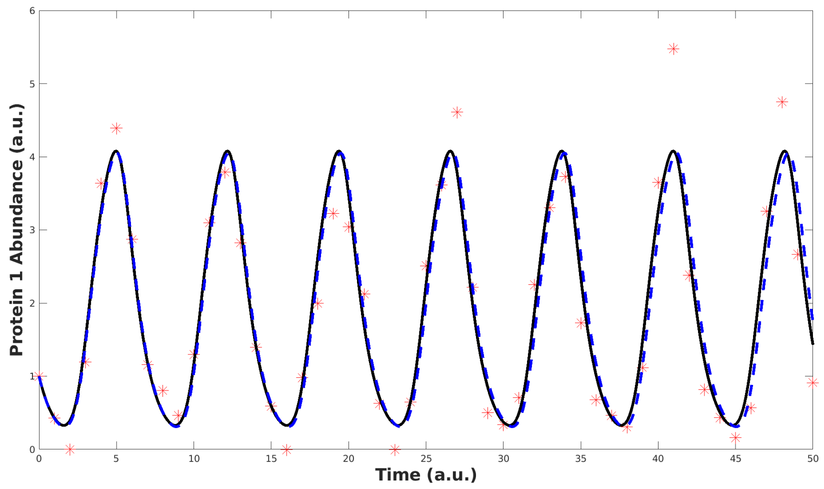
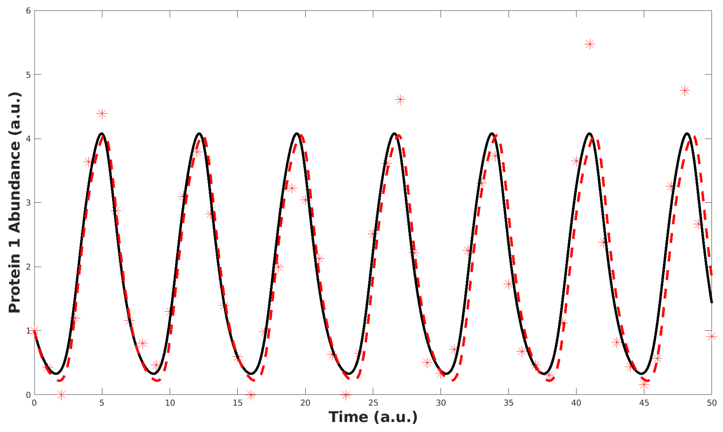


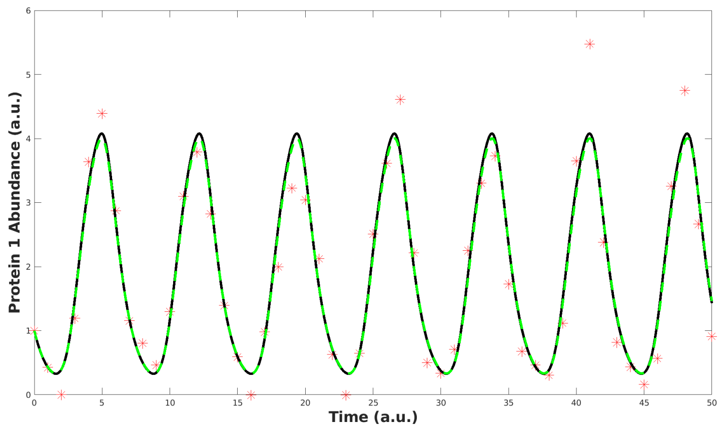
Figure 4.16: The red points plot the data set (1, 0.5). The black curve is the model solution of system (4.9). The dotted blue curve is the model solution of system (4.9) with the parameters given in Table 4.5.

Figure 4.21 plots the top parameter estimate when fitting the data set (1, 0.5). The black curve in Figure 4.21 is the model solution of system (4.9). From Figure 4.21, we see that the model solution given the parameter estimate found using Algorithm 4.6.2 varies little from the model solution of system (4.9).

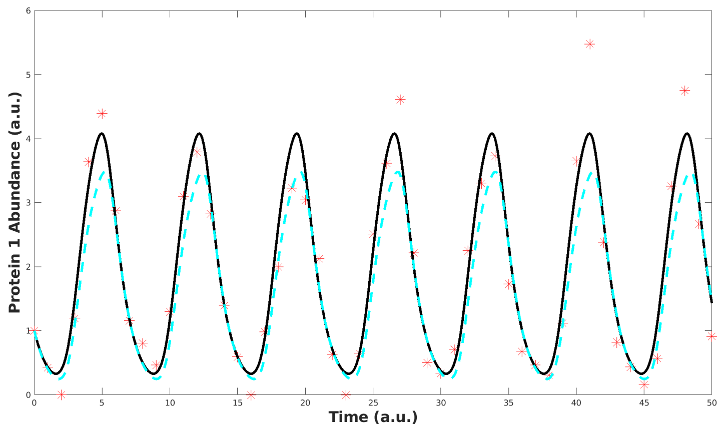
Finally, as in Section 4.7.1, we examine the variation in each parameter among the top 25 estimates of Algorithm 4.6.2. Figure 4.22 plots the frequency of the parameter estimates generated in Step 3 of Algorithm 4.6.2 (white histograms). The red histograms correspond to the top 25 parameter estimates using Algorithm 4.6.2. Similar to our findings in Section 4.7.1, we see from Figure 4.22 that there is relatively little variation in the top 25 estimates of the transcription rate and degradation ratio, which indicates that Algorithm 4.6.2 identifies these two parameters well. However, the top 25 estimates for the Hill coefficient vary from around 2.6 to as much as 7, indicating that the Hill coefficient is less identifiable than the other two parameters. Thus, we conclude that, from our possible test statistics considered in Algorithms 4.6.1 and 4.6.2, the cost of protein production is the best at discriminating among possible Hill coefficients.



(a)

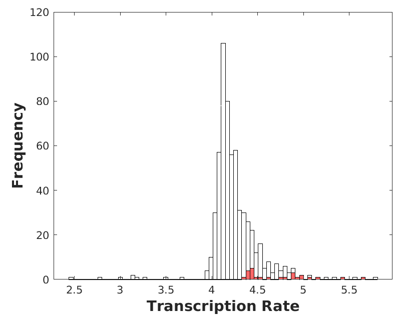


(b)

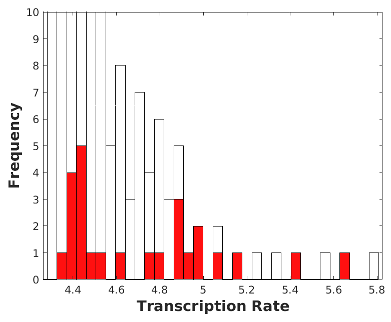


(c)

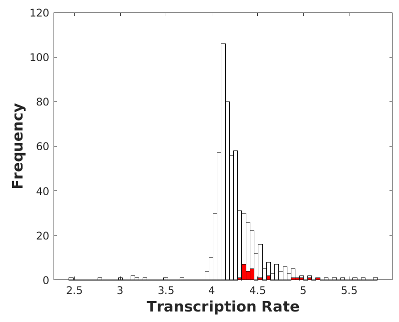
Figure 4.17: The red points in all three plot the data set (1, 0.5). The black curve in all three is the model solution of system (4.9). The dotted line is the model solution of system (4.9) with the parameters given by the estimates generated using Algorithm 4.6.1 when ranked by the (a) amplitude, (b) cost, and (c) period.



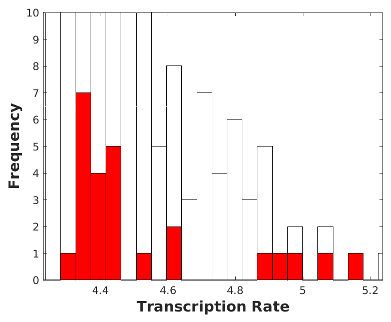
(a)



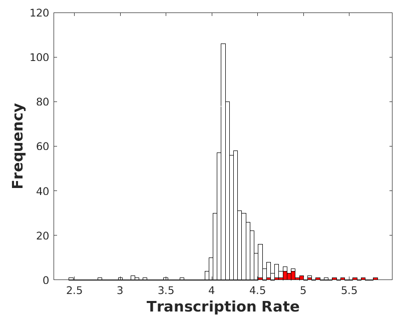
(e)



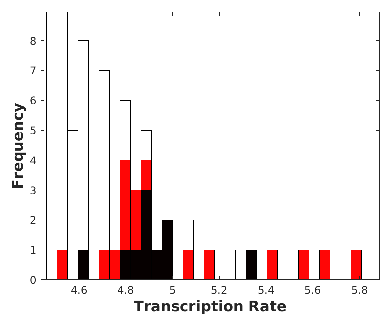
(b)



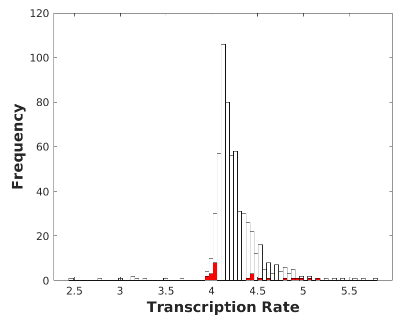
(f)



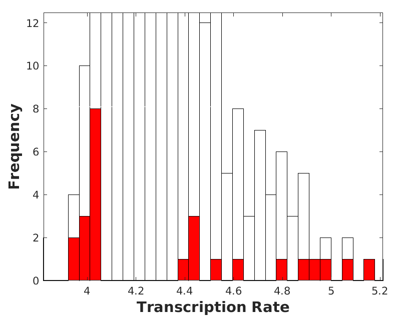
(c)



(g)

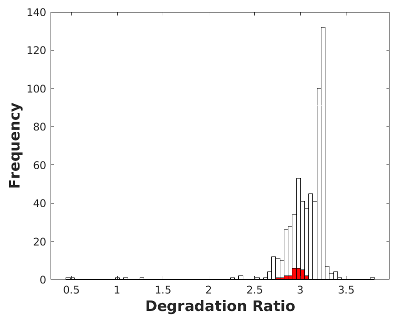


(d)

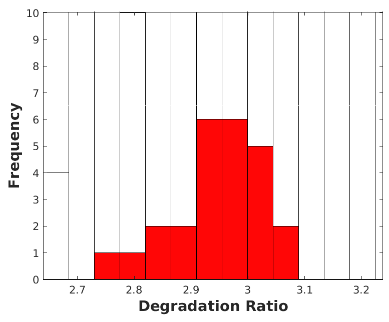


(h)

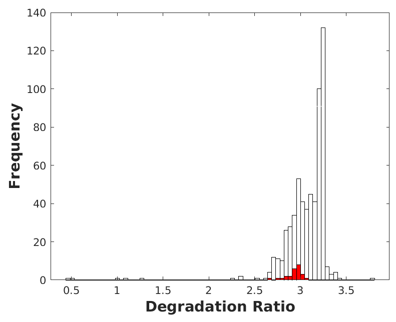
Figure 4.18: The white histogram for each of the above figures is the frequency of transcription rates generated from 600 repetitions of Step 4 in Algorithm 4.6.1. The red histograms are the top 25 transcription rates from Algorithm 4.6.1 when ranking by (a) sum of the three statistics, (b) the amplitude, (c) the cost, and (d) the period. Plots (e), (f), (g), and (h) are zoomed-in views of (a), (b), (c), and (d), respectively. The black histogram in (g) represents the top 10 transcription rates from Algorithm 4.6.1 when ranked by the cost.



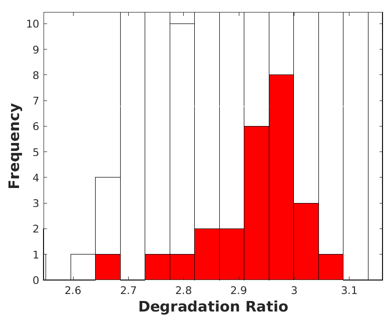
(a)



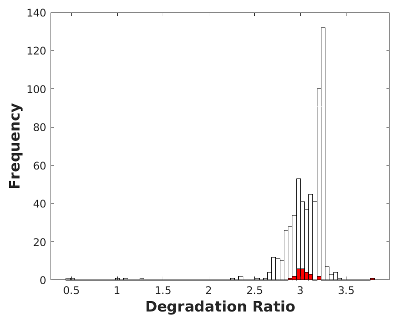
(e)



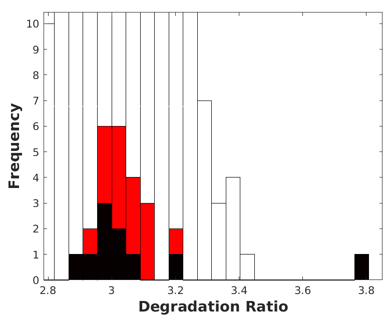
(b)



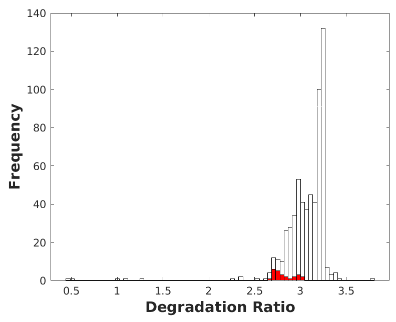
(f)



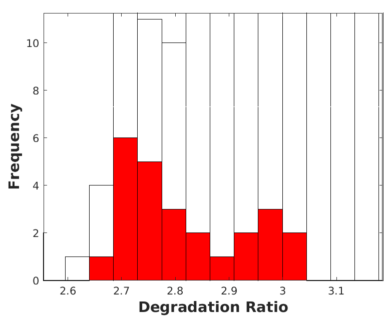
(c)



(g)



(d)



(h)

Figure 4.19: The white histogram for each of the above figures is the frequency of degradation ratios generated from 600 repetitions of Step 4 in Algorithm 4.6.1. The red histograms are the top 25 degradation ratios from Algorithm 4.6.1 when ranking by (a) sum of the three statistics, (b) the amplitude, (c) the cost, and (d) the period. Plots (e), (f), (g), and (h) are zoomed-in views of (a), (b), (c), and (d), respectively. The black histogram in (g) represents the top 10 degradation ratios from Algorithm 4.6.1 when ranked by the cost.

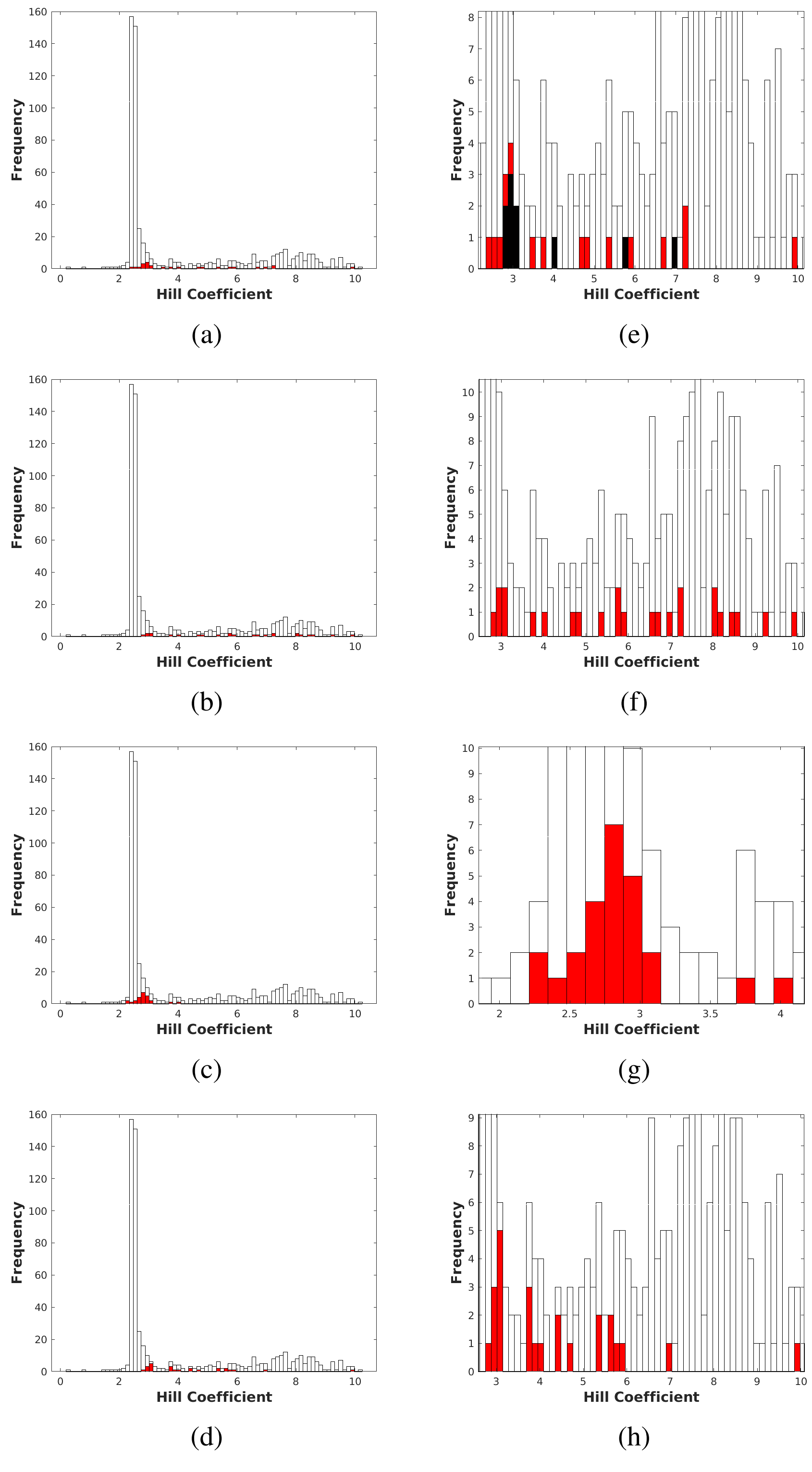


Figure 4.20: The white histogram for each of the above figures is the frequency of Hill coefficients generated from 600 repetitions of Step 4 in Algorithm 4.6.1. The red histograms are the top 25 Hill coefficients from Algorithm 4.6.1 when ranking by (a) sum of the three statistics, (b) the amplitude, (c) the cost, and (d) the period. Plots (e), (f), (g), and (h) are zoomed-in views of (a), (b), (c), and (d), respectively. The black histogram in (e) represents the top 10 Hill coefficients from Algorithm 4.6.1 when ranked by all three test statistics.

Data Set (s, σ)	TR (true = 5)	DR (true = 3)	Hill (true = 3)	Elapsed Time (minutes)
(0.5, 0)	5.02910172	2.99851551	2.95718491	18.25
(1, 0)	5.00142005	3.00017108	2.99859154	16.85
(0.5, 0.1)	5.02833460	2.99419699	2.91920914	18.39
(1, 0.1)	4.93001543	3.00347825	3.15639849	17.07
(0.5, 0.25)	5.14185394	3.00635434	2.85666843	18.78
(1, 0.25)	5.59612661	2.97756431	2.58264460	16.06
(0.5, 0.5)	4.76019839	2.96649153	3.48843037	18.49
(1, 0.5)	5.05740650	3.03809159	3.08531641	17.20

Table 4.7: Parameter Estimates generated by Algorithm 4.6.2 for the respective time spacing and noise added. Actual parameter values given in system (4.9) are 5, 3, and 3 for the transcription rate (TR), degradation ratio (DR), and Hill coefficient (Hill), respectively. The estimates colored red refer to parameter estimates that are worse than the corresponding estimate in Table 4.5.

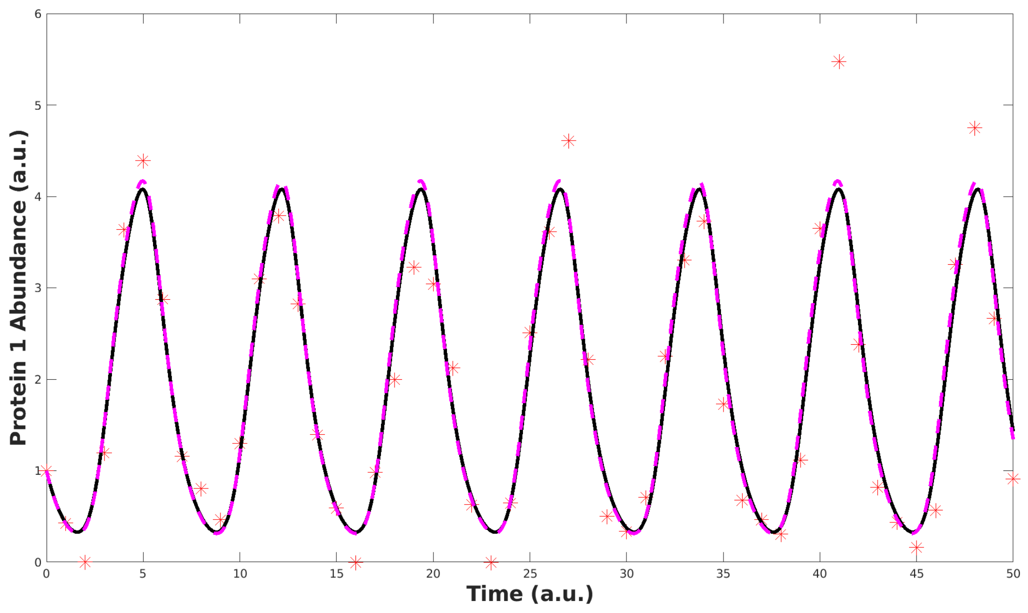
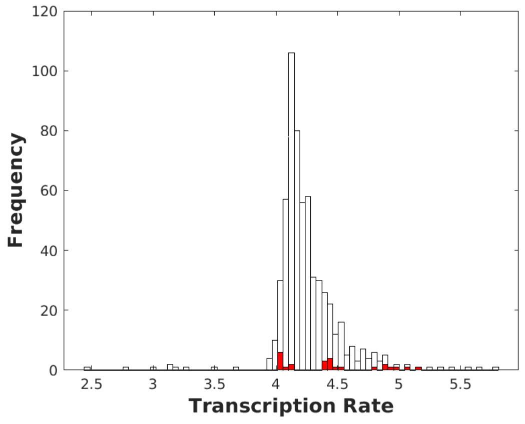
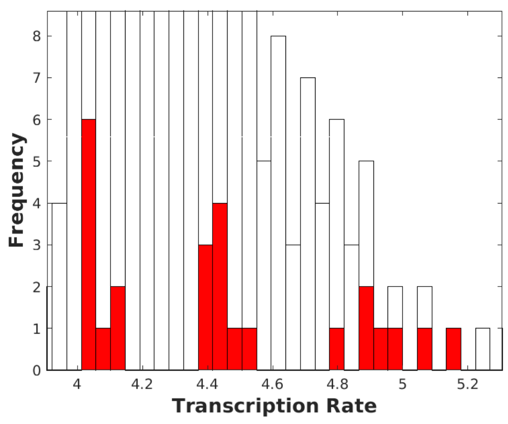


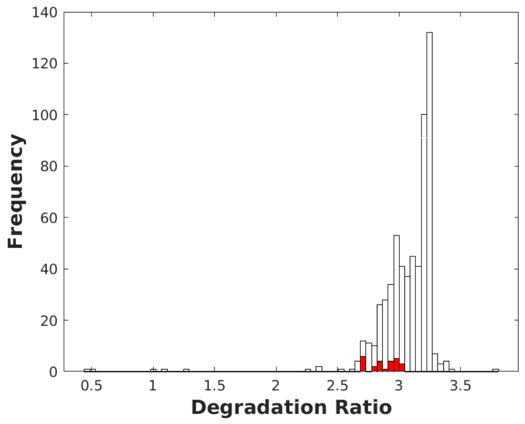
Figure 4.21: The red points are the data points from data set (1, 0.5). The black curve is the original model solution to system (4.9). The dotted curve is the model solution of system (4.9) with the parameters $\alpha = 5.0574$, $\beta = 3.0381$, and $n = 3.0853$ (Table 4.7, Row (1, 0.5)).



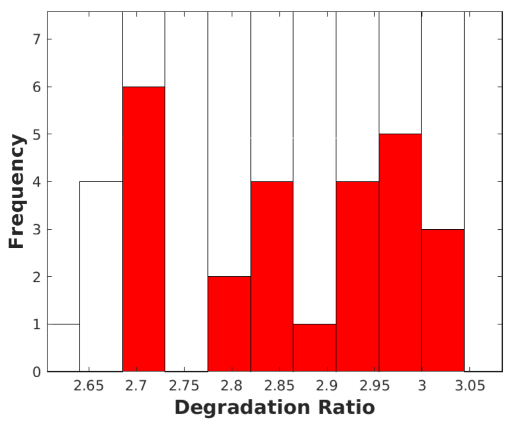
(a)



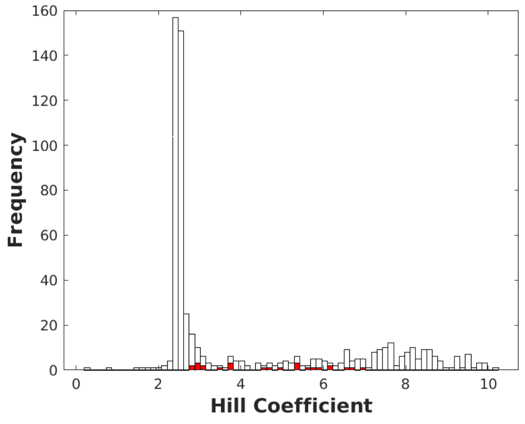
(d)



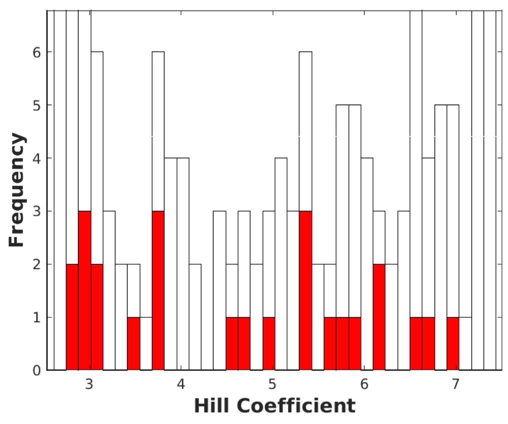
(b)



(e)



(c)



(f)

Figure 4.22: Results of Algorithm 4.6.2 on the data set (1, 0.5) (plotted in Figure 4.15(b)). The white histograms plot the frequency of the (a) transcription rates, (b) degradation ratios, and (c) Hill coefficients generated from 600 repetitions of Step 3 in Algorithm 4.6.2. The red histograms plot the frequency of the top 25 (a) transcription rates, (b) degradation ratios, and (c) Hill coefficients selected by Algorithm 4.6.2. Plots (d), (e), and (f) are zoomed-in plots of (a), (b), and (c), respectively. Recall that the true values are 5, 3, and 3 for the transcription rate, degradation ratio, and Hill coefficient, respectively.

Data Set	Parameter Estimate Means, Alg. 4.6.1			Parameter Estimate Means, Alg. 4.6.2		
(0.5, 0.5)	5.2426	2.9103	3.6912	4.9910	2.9655	3.4620
(1, 0.25)	5.0471	2.7693	4.3710	4.9770	2.9860	3.3266

Table 4.8: Mean transcription rate, degradation ratio, and Hill coefficient of the 100 runs of Algorithm 4.6.1 (Column 2) and Algorithm 4.6.2 (Column2) for the two data sets (Column 1).

4.7.3 Repeated Runs of Algorithms 4.6.1 and 4.6.2

To investigate the robustness of Algorithms 4.6.1 and 4.6.2, we run both algorithms on 100 of the (0.5, 0.5) and (1, 0.25) data sets simulated from system (4.9). In Figure 4.23, for each parameter, we plot the frequency of the Algorithm 4.6.1 and Algorithm 4.6.2 output. Figures 4.23(a) and 4.23(b) plot the frequency of the transcription rate output by Algorithms 4.6.1 and 4.6.2, respectively. Similarly, Figures 4.23(c) and 4.23(d) (Figures 4.23(e) and 4.23(f)) plot the frequency of the degradation ratios (Hill coefficients) selected by Algorithms 4.6.1 and 4.6.2, respectively. From Figure 4.23, we see that, for each parameter, the results of Algorithm 4.6.2 are significantly sharper about the actual parameter value.

In Table 4.8, we list the mean parameter estimates generated by Algorithm 4.6.1 (Column 2) and Algorithm 4.6.2 (Column 3) with respect to the data set (Column 1). Again, we see that the mean parameter estimates from Algorithm 4.6.2 are consistently closer to the actual parameter value (Table 4.8).

Furthermore, in Table 4.9, we list the number of times, for each algorithm, that each estimate deviated by more than ten percent of the actual value (for dataset (1, 0.25)). From Table 4.9, we see that, for all three parameters, Algorithm 4.6.2 yielded fewer estimates that deviated by more than ten percent of the actual value. Moreover, of the 45 times that Algorithm 4.6.2 generated an estimate outside of ten percent of the actual value, Algorithm 4.6.1 also yielded an estimate outside of ten percent of the actual value a total of 27 times (Table 4.9). In comparison, of the 127 times that Algorithm 4.6.1 output an estimate outside of ten percent of the actual value, Algorithm 4.6.2 only output an estimate outside of ten percent of the actual value a total of 27 times (Table 4.9).

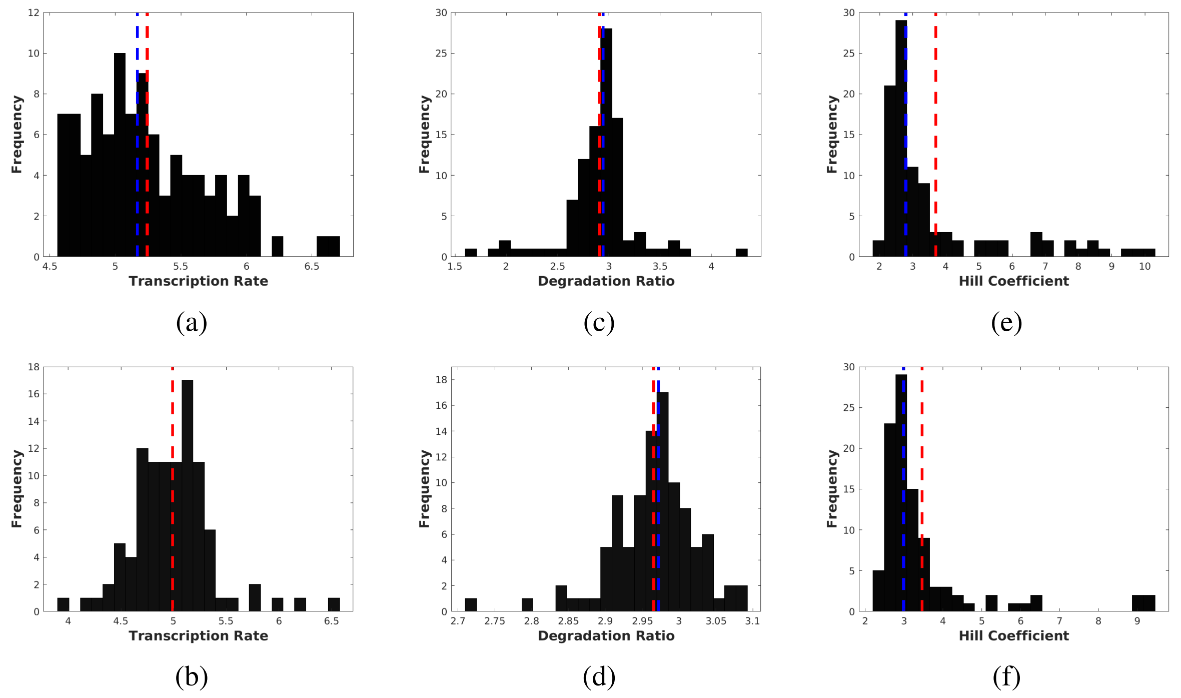


Figure 4.23: Algorithms 4.6.1 and 4.6.2 were run on 100 (0.5, 0.5) data sets. Plots (a) and (d) are the frequencies of transcription rates from Algorithms 4.6.1 and 4.6.2, respectively. Plots (b) and (e) are the frequencies of degradation ratios from Algorithms 4.6.1 and 4.6.2, respectively. Plots (c) and (g) are the frequencies of Hill coefficients from Algorithms 4.6.1 and 4.6.2, respectively. The red and blue lines in each plot are the mean and median values, respectively. Recall the actual parameter values of the transcription rate, degradation ratio, and Hill coefficient are 5, 3, and 3, respectively.

Algorithm	> 10% TR	> 10% DR	> 10% Hill
Algorithm 4.6.1	16 (3)	39 (0)	72 (24)
Algorithm 4.6.2	8 (3)	0	37 (24)

Table 4.9: Number of times the estimate from corresponding algorithm (Column 1) of the transcription rate (Column 2), degradation ratio (Column 3), and Hill coefficient (Column 4) deviated by more than ten percent of the actual value (5, 3, and 3, respectively). The numbers in parentheses count the number of times that the opposite algorithm's estimate also deviated by more than ten percent.

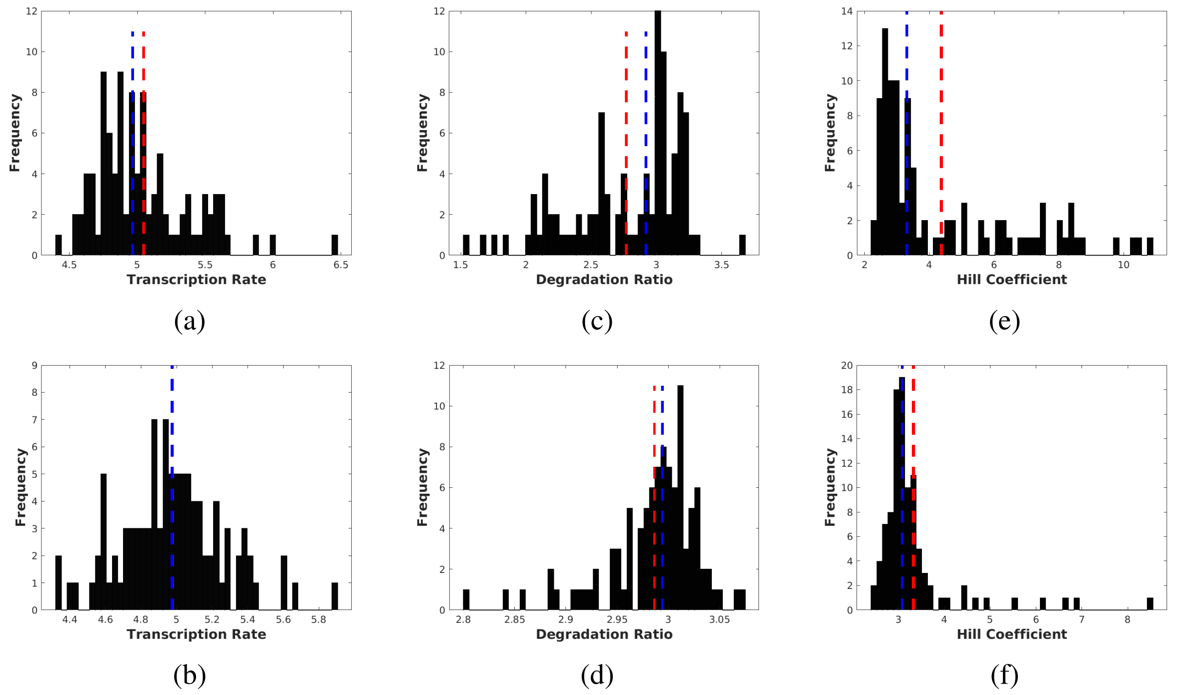


Figure 4.24: Algorithms 4.6.1 and 4.6.2 were run on 100 (1, 0.25) data sets. Plots (a) and (d) are the frequencies of transcription rates from Algorithms 4.6.1 and 4.6.2, respectively. Plots (b) and (e) are the frequencies of degradation ratios from Algorithms 4.6.1 and 4.6.2, respectively. Plots (c) and (g) are the frequencies of Hill coefficients from Algorithms 4.6.1 and 4.6.2, respectively. The red and blue lines in each plot are the mean and median values, respectively. Recall the actual parameter values of the transcription rate, degradation ratio, and Hill coefficient are 5, 3, and 3, respectively.

In summary, both algorithms, when given simulated data with various levels of noise as input, produce particularly accurate parameter estimates. In general, however, the second algorithm, Algorithm 4.6.2 generates more accurate parameter estimates than the first algorithm, Algorithm 4.6.1, and the second algorithm is more robust to noise in the data set. In the next section, we compare Algorithm 4.6.2 to three other algorithms in the field of computational biology and show that it is as accurate and faster.

4.8 Final Algorithm and Algorithm Comparison

In light of our empirical results in Section 4.7, we conclude that Algorithm 4.6.2 is better at estimating parameters of system (4.9) than Algorithm 4.6.1. In Section 4.8.1, we compare our Algorithm 4.6.2 to standards in the field including the previous Constrained HEKF Algorithm with the Variance Test (Section 4.2.1.2), the particle swarm algorithm (a global optimization procedure), and GEARs [6]. We show that our parameter estimates are comparable to, if not better than, estimates generated using these other algorithms, and our algorithm saves significant time.

4.8.1 Algorithm Comparison

Below, we compare Algorithm 4.6.2 against three leading algorithms: the particle swarm optimization procedure (Section 4.8.1.1), the GEARS Toolbox [6] (Section 4.8.1.2), and the Constrained HEKF Algorithm with Variance Test (Section 4.8.1.3). We show that the parameter estimates generated by Algorithm 4.6.2 are comparable to the estimates generated using the particle swarm algorithm, and Algorithm 4.6.2 saves significant time (Table 4.10).

4.8.1.1 Comparison with the Particle Swarm Algorithm

First, we compare Algorithm 4.6.2 with the particle swarm algorithm on eight data sets generated in the same way as described in Section 4.7. We find that Algorithm 4.6.2 is significantly faster than the global optimization procedure without sacrificing accuracy in any of the parameters.

To compare, we implement Algorithm 4.6.2 with a repetition number of 500 (Step 3 in Algorithm 4.6.2). To implement the particle swarm optimization procedure in MATLAB, we use the *particleswarm* function. We build an initial population matrix by first creating a 30×3 matrix

Data Set	Algorithm 4.6.2			Time (min)	Particle Swarm			Time (min)
(0.5, 0)	5.0291	2.9985	2.9571	11.68	4.9994	3.0001	3.0005	26.62
(1,0)	5.0015	3.0001	2.9983	14.53	5.0001	3.0000	2.9999	23.13
(0.5, 0.1)	5.0129	2.9917	2.939	12.16	5.0148	2.998	2.9760	17.48
(1, 0.1)	4.9497	2.9830	2.9705	12.27	5.0180	3.0012	2.9832	21.88
(0.5, 0.25)	5.0846	2.9668	2.7541	10.89	5.0484	2.9988	2.9130	25.59
(1, 0.25)	4.9370	3.0303	3.0200	11.69	4.9934	2.9938	2.9325	33.13
(0.5, 0.5)	5.4596	2.9732	**2.5935**	11.80	5.3433	2.9859	**2.6690**	31.54
(1, 0.5)	5.0338	3.0467	2.9892	11.78	5.2651	3.0006	2.7420	34.15

Table 4.10: Comparison of Algorithm 4.6.2 with the Particle Swarm Algorithm in MATLAB. See Section 4.8.1.1 for a description of how the *particleswarm* function was initialized and how the objective function was defined. The estimates in red deviate by more than ten percent of the true parameter value.

where each row is

$$[\max(p_i) \quad 10 \cdot \text{rand} \quad 10 \cdot \text{rand}].$$

Here, p_i represents the protein data, and rand is a uniformly distributed random number between 0 and 1. Then, we add normal noise with mean zero and standard deviation .25 to each entry in the initialized matrix. We constrain the *particleswarm* function with lower bounds of 0 for the transcription rate and degradation ratio and 2 for the Hill coefficient. We use upper bounds of 10 for each of the three parameters. Finally, we use Algorithm 4.6.3 to define an objective function based on the relative ℓ^2-norm error between the model solution values and the data values. **Code used for this comparison appears in Appendix A, Section A.2.11.**

Table 4.10 lists the results of the two algorithms and the time taken for each to finish. Overall, the parameter estimates given by Algorithm 4.6.2 are comparable to the estimates generated using the Particle Swarm Algorithm. The most inaccurate parameter estimate is the Hill coefficient–a fit of 2.5986 with an original value of 3–when fitting to data set (0.5, 0.5). However, the particle swarm estimate of 2.669 given the same data set is not significantly better. Moreover, the run time for Algorithm 4.6.2 is consistently less than half the run time for the Particle Swarm Algorithm.

Data Set	Algorithm 4.6.2			GEARS		
(0.5, 0)	4.9923	2.9985	3.0149	5.8894	2.5556	2.0423
(1,0)	5.0015	3.0001	2.9983	4.4322	3.5088	2.7834
(0.5, 0.1)	5.1077	2.9742	2.7770	5.1412	2.7624	2.3926
(1, 0.1)	4.8980	2.8735	3.2434	4.6304	2.7864	2.7787
(0.5, 0.25)	5.0890	2.9579	2.7172	4.9444	2.8049	2.4970
(1, 0.25)	4.9370	3.0303	3.0208	6.5915	3.3713	2.0750
(0.5, 0.5)	5.3476	2.9952	2.5986	4.8056	2.4742	2.1976
(1, 0.5)	5.0338	3.0467	2.9892	6.0895	2.0963	2.0000

Table 4.11: Comparison of Algorithm 4.6.2 and the GEARS Toolbox [6] given equivalent data sets. See Section 4.8.1.2 for a description of how we initialized and ran the GEARS Algorithm.

4.8.1.2 Comparison with the GEARS Toolbox

Next, we compare our algorithm with the optimization procedure in the GEARS Toolbox [6]. As in Section 4.8.1.1, we provide both algorithms with the same data sets. The GEARS algorithm requires an initialization of all mRNA and protein values. We initialize the protein values as the initial value in the data set. We initialize the mRNA values as $5 \cdot \text{rand}$ where rand is a uniformly distributed random number between 0 and 1. The GEARS Algorithm also requires a standard deviation value for each data point. For the data sets with zero noise, we set the standard deviation values as .1. For all other data sets, we set the standard deviation values as the standard deviation of the added noise. Finally, for the GEARS Algorithm, we set the lower bounds of the parameters at 10^{-3}, 10^{-3}, and 2 for the transcription rate, degradation ratio, and Hill coefficient, respectively. We set the upper bounds for the parameters as 10 for each.

Table 4.11 lists the parameter estimates generated from Algorithm 4.6.2 and the GEARS Algorithm. From Table 4.11, we see that all parameter estimates, with the exception of one, from Algorithm 4.6.2 are more accurate than the parameter estimate from the GEARS Algorithm. Also, although we do not report the times, the algorithms run in roughly the same amount of time.

Figure 4.25 plots one of the output plots from the GEARS Toolbox. The data set in Figure 4.25 is (0.5, 0.25). We choose this plot to report because the parameter estimate generated with this specific data set is the most accurate parameter estimate generated by GEARS.

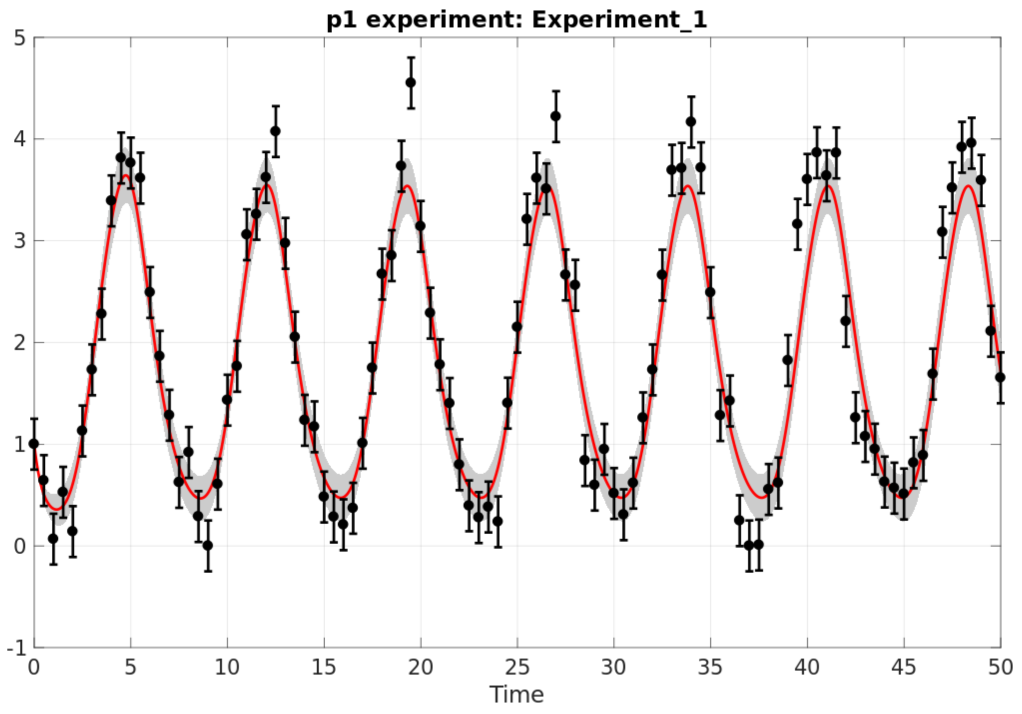


Figure 4.25: The Regularized Solution with Uncertainty plot generated by the GEARS Toolbox. The black data points are the data points from the data set (0.5, 0.25). We initialized the standard deviation of the data set in the GEARS Algorithm as .25 for each data point. The parameter estimate that GEARS generated was 4.9444, 2.8049, and 2.4970 with actual values of 5, 3, and 3, respectively.

4.8.1.3 Revisiting the Example in Section 4.4.1

Recall from Section 4.4.1 that we ran–for a total of 500 times–the Constrained HEKF Algorithm with the Variance Test on data set (1, 0.5) (Figure 4.5). The Variance Test passed ten parameter estimates (Table 4.2). However, the top ten most accurate parameter estimates–when ranked by the ℓ^1-norm difference between the actual parameter set and the parameter estimate–generated in the 500 iterations of the algorithm did not pass the Variance Test (Table 4.3). Here, we revisit this example and show that Algorithm 4.6.2 results in a more accurate parameter estimate than any of the estimates that passed the Variance Test.

To begin, we start at Step 4 of Algorithm 4.6.2 with the 500 parameter estimates that we generated in Section 4.4.1. The output of Algorithm 4.6.2 in this case was $\alpha = 4.91357316$, $\beta = 2.93884965$, and $n = 2.74271903$, where the actual values were $\alpha = 5$, $\beta = 3$, and $n = 3$. Recall that this specific parameter estimate was listed in Table 4.3 as one of the top-ten parameter estimates when ranked by the ℓ^1-norm error between the estimate and the actual values. Figure 4.26 plots the model solution with the new parameter estimate in magenta. Notice that the waveform of the new model solution closely follows that of the actual model solution (black curve) with the exception that the peak of the model solution with the new parameter estimate is slightly smaller than the peak of the actual model solution (Figure 4.26).

4.9 Discussion

Parameter estimation of the repressilator, and more generally biological oscillators, is important but challenging due to the nonlinearity present in the system. Specifically, we saw that estimating parameters of the repressilator involves two key challenges: multimodality [58] and lack of initial mRNA values. Previous attempts at parameter estimation of the repressilator either fail to address both issues or result in poor parameter estimates. For example, we showed that the GEARS Toolbox [6] requires all initial data for optimal results (Section 4.4.2). In addition, in Section 4.4.1, we show that the other standard procedure in the field, the Constrained HEKF Algorithm with Variance Test, pass parameter estimates that are less accurate than select estimates that were rejected.

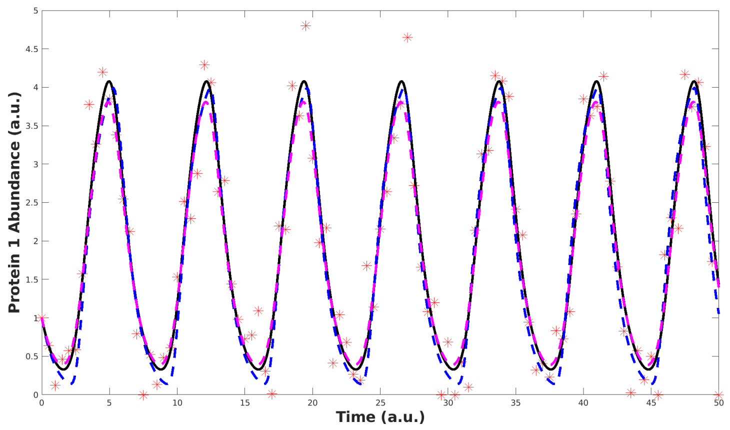


Figure 4.26: The red data points correspond to simulated data of system (4.9) with $\alpha_0 = 0$, $\alpha = 5$, $\beta = 3$, and $n = 3$. The data points were taken from the black curve at a spacing 1 time unit, and normal noise with mean zero and standard deviation .5 was added. The blue curve is the model solution of system (4.9) with a parameter set that passed the Variance Test from Section 4.4.1 (parameters $\alpha_0 = 0$, $\alpha = 4.6324$, $\beta = 2.9558$, and $n = 10.2211$). The magenta curve is the model solution with the parameter estimate generated from Algorithm 4.6.2 (parameters $\alpha = 4.91357316$, $\beta = 2.93884965$, and $n = 2.74271903$).

In this section, we present a novel procedure that addresses both issues and also incorporates information about the inherent oscillatory structure to generate more precise parameter estimates. We use as a foundation the Constrained HEKF Algorithm because it addresses the issue of multi-modality through its Bayesian approach. However, we seek a more informative test statistic than the variance. In Section 4.5, we investigate how various quantities (skewness, period, amplitude, cost of protein production, etc.) are sensitive to changes in the three parameters of the repressilator. We see that all quantities are sensitive to variations in the transcription rate and degradation ratio, but not necessarily the Hill coefficient. In fact, each quantity stabilizes after the Hill coefficient reaches a value between 5 and 10.

In light of our analysis in Section 4.5, we choose, as potential test statistics, three quantities: period, amplitude, and cost of protein production. These three quantities are as sensitive to variations in the parameters as the other quantities tested, and they are easy to estimate from a discrete data set. Using these quantities, in Section 4.6, we introduce two new algorithms to estimate the parameters of the repressilator. The first, Algorithm 4.6.1, generates many possible parameter estimates using the Constrained HEKF Algorithm and then ranks them based on how well the resulting system matches the period, amplitude, and cost of protein production of the data set.

Similarly, the second, Algorithm 4.6.2, computes many possible estimates but outputs the estimate with the smallest ℓ^2-norm between the model solution and the data points. To guarantee that our model solution values correspond to the time points of the data, we also outlined an event location procedure (Algorithm 4.6.3) that takes as input a parameter estimate and the time points (of the data set) and that outputs the model solution values coming from the input parameter estimate. Algorithm 4.6.3 can be coupled with an objective function inside another optimization procedure to yield more reliable parameter estimates.

In Section 4.7, we ran Algorithms 4.6.1 and 4.6.2 on eight data sets simulated from a simple repressilator model. We saw that, among the possible test statistics considered in Algorithm 4.6.1, simply ranking estimates using the cost of protein production or using the ℓ^2-norm error resulted in the most accurate parameter estimates. Thus, in Section 4.7.3, we further investigated the robustness of the two algorithms using 100 additional simulated data sets. Our analysis revealed that, in general, Algorithm 4.6.2 results in better parameter estimates.

Finally, continuing with Algorithm 4.6.2 as our final algorithm, in Section 4.8, we compare outputs of Algorithm 4.6.2 against outputs of the particle swarm algorithm (Section 4.8.1.1) and the GEARS Toolbox (Section 4.8.1.2). Algorithm 4.6.2 is as accurate as the particle swarm algorithm but in half the time. Also, Algorithm 4.6.2 is more accurate than the GEARS Toolbox in the same amount of time. Lastly, we revisit our example in Section 4.4.1 to show that Algorithm 4.6.2 gives a better parameter estimate than any that passed the original Variance Test.

In summary, we now have a parameter estimation procedure, Algorithm 4.6.2, that addresses the challenges of parameter estimation of biological systems that exhibit oscillations. The new algorithm is robust to noise in the system and is efficient in its implementation. The procedure also overcomes the lack of all initial data that is inherent to experimental data of the repressilator. Ultimately, robust parameter estimation procedures are crucial in faithfully representing biological systems to provide models with more predictive power. Therefore, an important future direction of our work is applying Algorithm 4.6.2 to actual repressilator data, e.g., data recently generated by the Paulsson lab at Harvard [63].

5. CONCLUSION AND FURTHER DIRECTIONS

In this dissertation, we advance the field of biological clocks through the rigorous analysis of a specific biological clock called the repressilator. Originally studied as a synthetic gene network, the repressilator led to a better functional understanding of transcriptional control by repression and also showed the promise of constructing, from naturally occurring components, artificial genetic networks with desirable functional properties [5]. Subsequently, the repressilator was identified as the core circadian mechanism of the plant circadian clock in *Arabidopsis thaliana* [3]. Here, we articulate our contributions to repressilator theory and the implications they have in the field of biological clocks, synthetic biology, and computational biology.

In Section 2, we develop and rigorously analyze the most general ODE model of the repressilator (System (2.7)). The model extends the work of Müller *et al.* by removing two biologically restrictive assumptions that seriously limit the model's applicability. Previous models used first-order degradation and translation functions with equivalent rates among the various mRNAs and proteins. In our generalization, we allow for monotone transcription-rate, translation-rate, and degradation-rate functions to model the respective processes. Our new system retains many advantageous qualitative properties of the previous repressilator after these generalizations. We analyze the system to find the number of possible steady states (Theorems 2.3.3, 2.3.7, 2.3.8, and 2.3.10), to characterize the stability of steady states (Theorems 2.3.13, 2.3.14, and 2.3.15), and to determine possible asymptotic behavior (Theorem 2.3.20). We also give a counterexample to Conjecture 1 in [21] (Section 2.3.3).

Taken together, our results from Section 2 advance the theoretical study of cyclic gene repression by generalizing the current repressilator models. We hope that our results will encourage theoretical and experimental biologists to broaden the possible degradation-rate and transcription-rate functions used to model the repressilator and other gene regulatory networks. Finally, we expect that allowing general functions for these terms will generate more accurate and predictive models of not only the repressilator but genetic repression in general.

In Section 3, we derive a new transcription-rate function (Eqn. (3.14)) that arises under more reasonable biological assumptions than the commonly used Hill function (see Remark 2.3.1). We then compare the qualitative properties of model solutions of the repressilator with our new transcription-rate function versus model solutions of the repressilator with the Hill function. From our numerical simulations, we see that the period, amplitude, and phase of oscillations vary drastically between the solutions of the two systems. In particular, with the successive-binding function, the period is more sensitive to the Hill coefficient and continues to increase with increasing Hill coefficient whereas the period stabilizes quickly with respect to the Hill coefficient given the model using the Hill function. We find that the phase difference behave similarly to the period. Protein abundance amplitudes, however, stabilize quickly for a model with the new transcription-rate function as opposed to those computed from a model with the Hill function as the transcription-rate function. These differences in dynamics are a result of the new transcription-rate function modeling a stronger repressive action than the previous Hill function.

In addition, we present a conjecture, with supporting theoretical motivation, that the amplitude of protein abundance of the repressilator model with the Hill function is always greater than the amplitude of protein abundance of the repressilator with our new transcription-rate function, all other parameters and functions being equal. For example, we proved that the transcription rate is an upper bound of limit cycle values of the repressilator model with the Hill function and the repressilator model with our new transcription-rate function (Theorems 3.4.1 and 3.4.2). In the end, we stated a conjecture comparing the amplitudes of the two models (Conjecture 3.4.4).

In Section 4, we present three novel algorithms to address the challenges of parameter estimation of models of biological oscillators. Two algorithms, when given time-course data of biological oscillators, output parameter estimates of the model (Algorithms 4.6.1 and 4.6.2). The foundation for the parameter estimation procedures was the Constrained HEKF Algorithm [57]. Instead of the Variance Test as an *a posteriori* identifiability test, we introduce new quantities to exploit for identifiability. For example, we use the period, amplitude, and cost of protein production to discriminate among possible parameter estimates. In the second algorithm, we select the parameter estimate that results in the least ℓ^2-norm error between the model solution given the parameter estimate and the data points. We ran both algorithms on simulated data sets and showed that our algorithms work well at recovering accurate parameter estimates. Moreover, we compare our algorithms to standard algorithms in the field of computational biology and show that our new algorithm works faster and more accurately than any that we test against.

Furthermore, we address the issue of the lack of initial mRNA values by producing a new algorithm that takes as input time-course data and a parameter estimate and outputs model solution values given that parameter estimate that aligns with the data points (Algorithm 4.6.3). We note that Algorithm 4.6.3 can be coupled with optimization procedures that minimize an objective function of relative-error between data and model solution values.

5.1 Further Directions

In this section, we highlight some further directions for our research.

5.1.1 Parameter Estimation of Actual Repressilator Data

In Section 4, we developed a novel algorithm to estimate parameters of models of biological clocks. Moreover, we applied it to parameter estimation of a repressilator model given simulated data. Next, we intend to use Algorithm 4.6.2 in Section 4 to fit actual clock data to uncover rates of transcription, degradation, and translation of the genes, proteins, and mRNAs, respectively.

Furthermore, an interesting future direction is applying Algorithm 4.6.2 to address the problem of *model selection*–identifying the mathematical model that most accurately reflects the actual biological mechanisms. For example, we will employ Algorithm 4.6.2 to select between a repressilator model with the Hill function modeling transcription and a corresponding repressilator model with our new transcription-rate function in Section 3. In developing a method to couple Algorithm 4.6.2 with a selection procedure, we will gain insight into the mechanisms of transcription, translation, and degradation.

5.1.2 A Generalized Stochastic Model of the Repressilator

In this dissertation, we were concerned with generalizing a deterministic ODE model of the repressilator. However, biological processes are inherently stochastic in nature. Thus, we will continue our work by translating our generalized repressilator system to a stochastic model. Also, we will analyze the dynamics of the stochastic system to uncover what, if any, advantages the stochastic system confers on the repressilator network.

REFERENCES

[1] M. Smolensky and L. Lamberg. *The Body Clock Guide to Better Health*. Henry Holt and Company, LLC, New York, New York 10011, 1st edition, 2001.

[2] M. B. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *NATURE*, 403, January 2000.

[3] A. Pokhilko, A. P. Fernández, K. D. Edwards, M. M. Southern, K. J. Halliday, and A. J. Millar. The clock gene circuit in arabidopsis includes a repressilator with additional feedback loops. *MOL SYST BIOL*, 8(1), 2012.

[4] Matlab optimization toolbox, R2017a. The MathWorks, Natick, MA, USA.

[5] M. B. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *NATURE*, 403:335–338, 2000.

[6] J. A. Pitt and J. R. Banga. Parameter estimation in models of biological oscillators: an automated regularised estimation approach. *BMC BIOINFORMATICS*, 20(1):82, Feb 2019.

[7] C. H. Johnson. Circadian clocks and cell division: what's the pacemaker? *CELL CYCLE (Georgetown, Tex.)*, 9(19):3864–3873, 10 2010.

[8] L. Ziv and Y. Gothilf. Circadian time-keeping during early stages of development. *PNAS*, 103(11):4146–4151, 2006.

[9] R. Lev Bar-Or, R. Maya, L. A. Segel, U. Alon, A. J. Levine, and M. Oren. Generation of oscillations by the p53-mdm2 feedback loop: A theoretical and experimental study. *PNAS*, 97(21):11250–11255, 2000.

[10] B. Novak and J. Tyson. Design principles of biochemical oscillators. *NAT REV MOL CELL BIO*, 9:981–991, December 2008.

[11] T. Y. Tsai, Y. S. Choi, W. Ma, J. R. Pomerening, C. Tang, and J. E. Ferrell. Robust, tunable biological oscillations from interlinked positive and negative feedback loops. *SCIENCE*, 321(5885):126–129, 2008.

[12] A. Goldbeter, C. Gérard, D. Gonze, J.-C. Leloup, and G. Dupont. Systems biology of cellular rhythms. *FEBS LETT*, 586(18):2955–2965.

[13] E. Gwinner. Circadian and circannual programmes in avian migration. *J EXP BIOL*, 199(1):39–48, 1996.

[14] K. Straif, R. Baan, Y. Grosse, B. Secretan, and F. El Ghissassi. Carcinogenicity of shift-work, painting, and fire-fighting. *LANCET ONCOL*, 8:1065–1066, December 2007.

[15] J. Hasty, F. Isaacs, M. Dolnik, D. McMillen, and J. J. Collins. Designer gene networks: Towards fundamental cellular control. *CHAOS*, 11(1):207–220, 2001.

[16] A. Goldbeter. Computational approaches to cellular rhythms. *NATURE*, 420:238–245, 2002.

[17] S. Becker-Weimann, J. Wolf, H. Herzel, and A. Kramer. Modeling feedback loops of the mammalian circadian oscillator. *J THEOR BIOL*, 87:3023–3034, 2004.

[18] J. C. Leloup and A. Goldbeter. Towards a detailed computational model for the mammalian circadian clock. *PNAS*, 100:7051–7056, 2003.

[19] H. Huang and D. A. Nusinow. Into the evening: Complex interactions in the arabidopsis circadian clock. *TRENDS GENET*, 32(10):674 – 686, 2016.

[20] S. Müller, J. Hofbauer, L. Endler, C. Flamm, S. Widder, and P. Schuster. A generalized model of the repressilator. *J MATH BIOL*, 53(6):905–937, Dec 2006.

[21] J Tyler, A Shiu, and J Walton. Revisiting a synthetic intracellular regulatory network that exhibits oscillations. *J MATH BIOL*, 78:2341–2368, 2019.

[22] M. A. Savageau. Design principles for elementary gene circuits: Elements, methods, and examples. *CHAOS*, 11(1):142–159, 2001.

[23] J. K. Kim. Protein sequestration versus Hill-type repression in circadian clock models. *IET SYST BIOL*, 10(4):125–135, July 2016.

[24] B. Novák and J. J. Tyson. Design principles of biochemical oscillators. *NAT REV MOL CELL BIO*, 9, 10 2008.

[25] N. Barkai and S. Leibler. Circadian clocks limited by noise. *NATURE*, 403, 01 2000.

[26] JM. G. Vilar, H. Y. Kueh, N. Barkai, and S. Leibler. Mechanisms of noise-resistance in genetic oscillators. *PNAS*, 99(9):5988, 04 2002.

[27] P. François and V. Hakim. Core genetic module: The mixed feedback loop. *PHYS REV E*, 72(3):031908–, 09 2005.

[28] J. K. Kim, D. B. Forger, M. Marconi, D. Wood, A. Doran, T. Wager, C. Chang, and K. M. Walton. Modeling and validating chronic pharmacological manipulation of circadian rhythms. *CPT: Pharmacometrics & Systems Pharmacology*, 2(7):57, 2018/10/15 2013.

[29] J. K. Kim and D. B Forger. A mechanism for robust circadian timekeeping via stoichiometric balance. *MOL SYST BIOL*, 8(1), 01 2012.

[30] Q. He, H. Shu, P. Cheng, S. Chen, L. Wang, and Y. Liu. Light-independent Phosphorylation of WHITE COLLAR-1 regulates its function in the Neurospora circadian negative feedback loop. *J BIOL CHEM*, 280:17526–17532, April 2005.

[31] Q. He and Y. Liu. Molecular mechanism of light responses in neurospora: from light-induced transcription to photoadaptation. *GENES DEV*, 19(23):2888–2899, 12 2005.

[32] Q. He, J. Cha, Q. He, H.-C Lee, Y. Yang, and Y. Liu. CKI and CKII mediate the frequency-dependent phosphorylation of the WHITE COLLAR complex to close the Neurospora circadian negative feedback loop. *GENES DEV*, 20(18):2552–2565, 09 2006.

[33] T. Schafmeier, A. Haase, K. Káldi, J. Scholz, M. Fuchs, and M. Brunner. Transcriptional feedback of Neurospora circadian clock gene by phosphorylation-dependent inactivation of its transcription factor. *CELL*, 122(2):235–246, 2005.

[34] G. Huang, S. Chen, S. Li, J. Cha, C. Long, L. Li, Q. He, and Y. Liu. Protein kinase a and casein kinases mediate sequential phosphorylation events in the circadian negative feedback loop. *GENES DEV*, 21(24):3283–3295, 12 2007.

[35] X. Yang. Generalized form of Hurwitz-Routh criterion and Hopf bifurcation of higher order. *APPL MATH LETT*, 15:615–621, 2002.

[36] L. J. S. Allen. *An Introduction to Mathematical Biology*. Pearson, Upper Saddle River, NJ, 2006.

[37] J. D. Murray. *Mathematical Biology*, volume 19 of *Biomathematics*. Springer-Verlag, Berlin Heidelberg, second, corrected edition, 1993.

[38] J. Mallet-Paret and H. L. Smith. The Poincaré-Bendixson theorem for monotone cyclic feedback systems. *J DYN DIFFER EQU*, 2(4):367–421, Oct 1990.

[39] S.M. Berget, C. Moore, and P. A. Sharp. Spliced segments at the 5' terminus of adenovirus 2 late mRNA. *PNAS*, 74(8):3171–3175, August 1977.

[40] O. I. Kulaeva, F. Hsieh, H. Chang, D. S. Luse, and V. M. Studitsky. Mechanism of transcription through a nucleosome by RNA polymerase II. *BIOCHIM BIOPHYS ACTA*, 1829(1):76–83, January 2013.

[41] G. M. Cooper. *The Cell: A Molecular Approach*. Sinauer Associates, Sunderland, MA, 2000.

[42] K. M. Page and R. Perez-Carrasco. Degradation rate uniformity determines success of oscillations in repressive feedback regulatory networks. *J R SOC INTERFACE*, 15(142), 2018.

[43] Q. He and Y. Liu. Degradation of the neurospora circadian clock protein frequency through the ubiquitin–proteasome pathway. *BIOCHEM SOC T*, 33(5):953–956, 2005.

[44] B. C. Goodwin. Oscillatory behavior in enzymatic control processes. *ADV ENZYME REGUL*, 3:425–438, 1965.

[45] D. B. Forger. *Biological Clocks, Rhythms, and Oscillations: The Theory of Biological Timekeeping*. The MIT Press, Cambridge, MA, 2017.

[46] D. Ullrich. *Complex Made Simple*. American Mathematical Society, Providence, Rhode Island, 2008.

[47] T. Mori and C. H. Johnson. Circadian programming in cyanobacteria. *Seminars in Cell & Developmental Biology*, 12(4):271–278, 2001.

[48] T. Kondo. A cyanobacterial circadian clock based on the kai oscillator. *COLD SPRING HARB SYM*, 72:47–55, 2007.

[49] J. S. Takahashi. Molecular neurobiology and genetics of circadian rhythms in mammals. *ANNU REV NEUROSCI*, 18(1):531–553, 1995. PMID: 7605073.

[50] S. M. Reppert and D. R. Weaver. Coordination of circadian timing in mammals. *NATURE*, 418(6901):935–941, 2002.

[51] J. S. Griffith. Mathematics of cellular control processes I. Negative feedback to one gene. *J THEOR BIOL*, 20:202–208, 1968.

[52] D. Gonze and W. Abou-Jaoudé. The Goodwin model: Behind the Hill function. *PLOS ONE*, 8(8):e69573–, 08 2013.

[53] L. Segel and M. Slemrod. The quasi-steady-state assumption: A case study in perturbation. *SIAM REV*, 31(3):446–477, 1989.

[54] O. Veblen. Theory on plane curves in non-metrical analysis situs. *TRANS AMER MATH SOC*, 6:83–98, 1905.

[55] N. Strelkowa and M. Barahona. Transient dynamics around unstable periodic orbits in the generalized repressilator model. *CHAOS*, 21(2):023104, 2011.

[56] I Potapov, B. Zhurov, and E. Volkov. Multi-stable dynamics of the non-adiabatic repressilator. *J R SOC INTERFACE*, 12(104):20141315, 2015.

[57] G. Lillacci and M. Khammash. Parameter estimation and model selection in computational biology. *PLOS COMPUT BIOL*, 6(3):1–17, 03 2010.

[58] A. Villaverde and J. Banga. Reverse engineering and identification in systems biology: strategies, perspectives, and challenges. *J R SOC INTERFACE*, 11, 2013.

[59] H. Jo, Y. J. Kim, J. K. Kim, M. Foo, D. E. Somers, and P. Kim. Waveforms of molecular oscillations reveal circadian timekeeping mechanisms. *NAT COMMUN BIOL*, 1(1):207, 2018.

[60] D. Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley-Interscience, New York, NY, USA, 2006.

[61] E. Ziegel. *Kendall's Advanced Theory of Statistics, Vol. 1: Distribution Theory*, volume 31. Taylor & Francis, 1989.

[62] J. Kim and D. Forger. On the existence and uniqueness of biological clock models matching experimental data. *SIAM J APPL MATH*, 72(6):1842–1855, 2012.

[63] L. Potvin-Trottier, N. D. Lord, G. Vinnicombe, and J. Paulsson. Synchronous long-term oscillations in a synthetic gene circuit. *NATURE*, 538:514–517, 10 2016.

APPENDIX A

SELECT MATLAB CODE

A.1 Section 3 Codes

A.1.1 Estimating Hopf Bifurcations of Models (SS) and (SB)

```
%%% Estimate the Hopf Bifurcations %%%

% Initialize the symbolic variables

syms x h p y

% Save the Jacobian

J = [-1 0 0 0 0 x;
    0 -1 0 x 0 0;
    0 0 -1 0 x 0;
    3 0 0 -3 0 0;
    0 3 0 0 -3 0;
    0 0 3 0 0 -3];

% Compute the characteristic polynomial

p = charpoly(J);

% Compute H5

H5 = [p(2) p(1) 0 0 0;
    p(4) p(3) p(2) p(1) 0;
    p(6) p(5) p(4) p(3) p(2);
    0 p(7) p(6) p(5) p(4);
    0 0 0 p(7) p(6)];

% Compute the determinant

D5 = det(H5);

% Find the negative, real zero of the determinant
```

```
sols = solve(D5);

r1 = double(sols(1));

%% Fix h = 3

% SB Model Hopf bifurcation with respect to the transcription rate

sols_p = solve([y== k/(1+y)^3, a == -3*k*(1+y)^(2)/(1+y)^6]);

k_sb = double(sols_p.k(1));

% SS Model Hopf bifurcation with respect to the transcription rate

sols_p = solve([y== k/(1+y^3), a == -3*k*(y)^(2)/(1+y^3)^2]);

k_ss = double(sols_p.k(1));

%% Fix k = 10

% SB Model Hopf bifurcation with respect to the Hill coefficient

sols_p = solve([y== k/(1+y)^3, a == -3*k*(1+y)^(2)/(1+y)^6]);

h_sb = double(sols_p.h(1));

% SS Model Hopf bifurcation with respect to the Hill coefficient

sols_p = solve([y== 10/(1+y^h), 1 == -h*10*y^(h-1)/(1+y^h)^2])

h_ss = double(sols_p.h(1));
```

A.1.2 Computing Phase, Amplitude, and Periods

Below, I furnish the code (*PhAP_diff.m*) that I wrote to illustrate the differences in phase, amplitude, and period between a repressilator model with the Hill function and the corresponding repressilator model with the successive-binding transcription-rate function. This code was used to generate figures in Section 3.

```
% PhAP - Phase, Amplitude, and Period Comparison between two models: one with Hill
% function and the other with successive-binding assumption

% How does phase vary with respect to alpha first?

alpha = 2:0.2:50;

phase_diffs = [];
periods     = [];
amplitudes  = [];

phase_diffs_b = [];
periods_b     = [];
amplitudes_b  = [];


for a = alpha

    % Parameter set [transcription-rate degradation-ratio Hill-coefficient alpha_0]

    p = [a 3 3 0];

    % Compute phase difference between protesins 1 and 2

    % First, run to limit cycle (if it stabilizes to a limit cycle)

    % Use the event locator to find when the protein 1 level peaks

    options = odeset('Events', @initial_events);

    [ts, xs, te, ye, ie] = ode23tb(@ode_repress2, [0 3000], [10 0 10 0 10 10 30 30], options, p);

    [tsb, xsb, teb, yeb, ieb] = ode23tb(@ode_repress2_sb, [0 3000], [10 0 10 0 10 10 30 30], options, p(1:3));


    % Calculate phase diff

    if(length(te)< 10)
        phase_diff_temp = 0;
        amp_temp        = 0;
        period_temp     = 0;
```

```
elseif((te(end)-te(end-4)) - (te(end-4)-te(end-8)) < .01)

    % Period Estimate

    period_temp = te(end)-te(end-4);

    % Estimate Amplitude diff

    for i = 0:3
        if(ye(end-i, 8) < 0 &ye(end-i,8) > -.001)
            p2max = i;
        elseif(ye(end-i,8) > 0 & ye(end-i,8) <.001)
            p2min = i;
        elseif(ye(end-i,7) < 0 & ye(end-i,7) > -.001)
            p1max = i;
        else
            p1min = i;
        end
    end

    amp_temp = ye(end-p1max,2) - ye(end-p1min,2);
    if(p2max > p1max)
        phase_diff_temp = te(end-p2max) - te(end-p1max) + period_temp;
    else
        phase_diff_temp = te(end-p2max) - te(end-p1max);
    end

else
    phase_diff_temp = 0;
    amp_temp        = 0;
    period_temp     = 0;
end

% Save the phase, amplitude, and period

phase_diffs = [phase_diffs phase_diff_temp];
periods     = [periods period_temp];
amplitudes  = [amplitudes amp_temp];

if(length(teb)< 10)
    phase_diff_temp_b = 0;
```

```
        amp_temp_b        = 0;
        period_temp_b     = 0;
    elseif((teb(end)-teb(end-4)) - (teb(end-4)-teb(end-8)) < .01)

        % Period Estimate

        period_temp_b = teb(end)-teb(end-4);

        % Estimate Amplitude diff

        for i = 0:3
            if(yeb(end-i, 8) < 0 &yeb(end-i,8) > -.001)
                p2max = i;
            elseif(yeb(end-i,8) > 0 & yeb(end-i,8) <.001)
                p2min = i;
            elseif(yeb(end-i,7) < 0 & yeb(end-i,7) > -.001)
                p1max = i;
            else
                p1min = i;
            end
        end

        amp_temp_b = yeb(end-p1max,2) - yeb(end-p1min,2);
        if(p2max > p1max)
            phase_diff_temp_b = teb(end-p2max) - teb(end-p1max) + period_temp_b;
        else
            phase_diff_temp_b = teb(end-p2max) - teb(end-p1max);
        end

    else
        phase_diff_temp_b = 0;
        amp_temp_b        = 0;
        period_temp_b     = 0;
    end

    phase_diffs_b = [phase_diffs_b phase_diff_temp_b];
    periods_b     = [periods_b period_temp_b];
    amplitudes_b  = [amplitudes_b amp_temp_b];
end

figure
```

```
plot(alpha, phase_diffs, 'k', 'LineWidth', 3)
hold on
plot(alpha, phase_diffs_b, 'r', 'LineWidth', 3)
xlabel('Hill Coefficient', 'FontSize', 14, 'FontWeight', 'bold')
ylabel('Phase Difference', 'FontSize', 14, 'FontWeight', 'bold')
hold off
figure
plot(alpha, amplitudes, 'k', 'LineWidth', 3)
hold on
plot(alpha, amplitudes_b, 'r', 'LineWidth', 3)
xlabel('Hill Coefficient', 'FontSize', 14, 'FontWeight', 'bold')
ylabel('Amplitude', 'FontSize', 14, 'FontWeight', 'bold')
hold off
figure
plot(alpha, periods, 'k', 'LineWidth', 3)
hold on
plot(alpha, periods_b, 'r', 'LineWidth', 3)
xlabel('Hill Coefficient', 'FontSize', 14, 'FontWeight', 'bold')
ylabel('Period', 'FontSize', 14, 'FontWeight', 'bold')
hold off

function [value, isterminal, direction] = initial_events(t,x,p)
value = [x(7); x(8)];
isterminal = [0;0];
direction = [0; 0];
end
```

In the above code, I call on two functions that give the ODEs for the two systems. I include the two functions below.

```
function dxdt = ode_repress2(t,x,p)
% The eight ODEs for the repressilator system with first-order degradation and translation.
% The transcription-rate functions are given by Hill functions.
% The two extra ODEs at the end are the second derivatives of $\dot{p}_1$ and $\dot{p}_2$,
% respectively.  They allow for the identification of the peaks and troughs of the protein abundance.
dxdt = [p(4)+p(1)/(1+x(6)^p(3)) - x(1); p(2)*(x(1)-x(2)); p(4)+p(1)/(1+x(2)^p(3)) - x(3);
p(2)*(x(3)-x(4)); p(4)+p(1)/(1+x(4)^p(3)) - x(5); p(2)*(x(5)-x(6));
p(2)*(p(1)/(1+x(6)^p(3)) - x(1) - p(2)*(x(1)-x(2)));p(2)*(p(1)/(1+x(2)^p(3)) - x(3) - p(2)*(x(3)-x(4))) ];
end

function dxdt = ode_repress2_sb(t,x,p)
% The eight ODEs for the repressilator system with first-order degradation and translation.
% The transcription-rate functions are given by the successive-binding transcription-rate function..
% The two extra ODEs at the end are the second derivatives of $\dot{p}_1$ and $\dot{p}_2$,
% respectively.  They allow for the identification of the peaks and troughs of the protein abundance.
dxdt = [p(1)/(1+x(6))^p(3) - x(1); p(2)*(x(1)-x(2)); p(1)/(1+x(2))^p(3) - x(3); p(2)*(x(3)-x(4));
p(1)/(1+x(4))^p(3) - x(5); p(2)*(x(5)-x(6)); p(2)*(p(1)/(1+x(6))^p(3) - x(1) - p(2)*(x(1)-x(2))) ;
p(2)*(p(1)/(1+x(2))^p(3) - x(3) - p(2)*(x(3)-x(4)))];
end
```

A.1.3 Plotting Limit Cycles

```
%%% Plot the Limit Cycles (Chapter 3) %%%

% Initialize the parameters

p = [9 3 5];

% Initialize the time span and initial conditions

tspan = [0 1000];
initials = [10 0 10 0 10 10];

% Run the ODE solver

[tss xss] = ode23tb(@ode_repress2, tspan, initials, [], p);

[tsb xsb] = ode23tb(@ode_repress2_sb, tspan, initials, [], p);

% Restrict the time

timess = find(tss > 900);
timesb = find(tsb > 900);

% Plot the limit cycles for mRNA1 and protein1

plot(xss(timess, 1), xss(timess,2), 'k', 'LineWidth', 3)
hold on
plot(xsb(timesb, 1), xsb(timesb,2), 'r', 'LineWidth', 3)
hold off
xlabel{'mRNA 1Abundance (a.u.)', 'FontSize', 14, 'FontWeight', 'bold')
ylabel('Protein 1 Abundance (a.u.)', 'FontSize', 14, 'FontWeight', 'bold')
```

A.2 Section 4 Codes

A.2.1 Constrained HEKF Algorithm

Below, I give the code used to compute the Kalman filter estimates of the repressilator parameters using the algorithm outlined by Khammash and Lilacci in [57].

```
%%%%%%%% EKF Algorithm on Simple Repressilator Models%%%%%%%%

% A program that uses the Constrained EKF Algorithm along with a skewness
% test to estimate parameters of an ODE System.

% The key steps to the algorithm are:
% 1. Load the data.
% 2. Initialize the system.
% 3. Compute the Jacobians at the previous a posteriori estimate.
% 4. Advance to the next time step (solve the ODE).
% 5. Compute the gain matrix.
% 6. Incorporate the current measurement to correct the prediction step.
% 7. Check if the estimates satisfy constraints on the states and
% parameters.
% 8. Repeat steps 3-7 for each time point in the dataset.

% States of the system: 3 mRNA, 3 protein, 3 parameter (TR, DR, Hill) = 9 total states

global p1_init p1_direction

%% Load the data

run('data_gen1.m');

%% Open the file

fileID = fopen('Parameters_4.txt', 'w');

%% Initialize the system.

% Initialize the nine states
r1_init = 10*rand;
p1_init = p1_data(1);
```

```
r2_init = 10*rand;
p2_init = p2_data(1);
r3_init = 10*rand;
p3_init = p3_data(1);
par1_init = 10*rand;
par2_init = 10*rand;
par3_init = 10*rand;

x0 = [r1_init
    p1_init
    r2_init
    p2_init
    r3_init
    p3_init
    par1_init
    par2_init
    par3_init];

% Initialize the error covariance matrix P

P0 = eye(9);
P0(2,2) = 2;%var(p1_data);
P0(4,4) = 2;%var(p2_data);
P0(6,6) = 2;%var(p3_data);

% Initialize the Q, R, and H matrices

Q = 5*eye(9);
R = eye(3);
R(1,1) = var(p1_data);
R(2,2) = var(p2_data);
R(3,3) = var(p3_data);
H = zeros(3,9);
H(1,2) = 1;
H(2,4) = 1;
H(3,6) = 1;

H_t = transpose(H);

% Initialize the constraint matrix
```

```
D = -1*eye(9);
d = zeros(9,1);

% Initialize estimates

[num_data, ~] = size(t_data);
p1_estimates = [p1_init];
p2_estimates = [p2_init];
p3_estimates = [p3_init];

%% Run the Algorithm



for i = 2:num_data
  % Compute the prediction term for x and P
  tspan = [t_data(i-1) t_data(i)];
        x_init_temp = x0(1:6);
         p = x0(7:end);
  [~, x] = ode23tb(@ode_repress2, tspan, x_init_temp, [], p);

          %Compute the new P
          P0 = newP_TR_DR_H(x0, P0, Q, tspan);

          x_temp = transpose(x(end,:));
          x_temp(7) = x0(7);
          x_temp(8) = x0(8);
          x_temp(9) = x0(9);

          % Compute the correction terms
  L = P0*H_t/(H*P0*H_t + R);
        residual = transpose(p_data(i,:)) - x_temp([2 4 6]);

          % Compute the new estimate
  x0 = x_temp + L*(transpose(p_data(i,:)) - x_temp([2 4 6]));
       p1_estimates = [p1_estimates x0(2)];
       p2_estimates = [p2_estimates x0(4)];
       p3_estimates = [p3_estimates x0(6)];

         % Compute corrected P
 P0 = (eye(9,9) - L*H)*P0*transpose(eye(9,9)-L*H) + L*R*transpose(L);
```

```
     % Test to see if we need to run the constraint
     test = find(x0 < 0);

     if(length(test > 0))
         fun = @(x) transpose(x-x0)*inv(P0)*(x-x0);
         x = fmincon(fun, x0, D, d)
         % Set the new x0
         x0 = x;
     end

end

parameter_estimate = x0(7:end)

fprintf(fileID, '%f \t %f \t %f \n', parameter_estimate)
```

A.2.1.1 Code to Update $P0$

In the above script for the HEKF Algorithm, we call a function called *newP_TR_DR_H* to update the error-covariance matrix, P. We provide the code below.

```
%% Matlab function that returns the new P matrix after solving the differential equation

function P = newP_TR_DR_H(x, P0, Q, tspan)
    A = f_Jac_TR_DR_H(x);

[~, P] = ode23tb(@(t,P)odefun(t,P,A,Q), tspan, P0(:));
P = P(end,:);
P = reshape(P, size(A));

end

function dPdt = odefun(t,P,A,Q)
    P = reshape(P, size(A));
  dPdt = A*P + P*transpose(A)+Q;
dPdt = dPdt(:);

end
```

A.2.1.2 Code to Compute the Jacobian

In the above script for *newP_TR_DR_H* , we call a function *f_Jac_TR_DR_H*. We provide the code below.

```
function A = f_Jac_TR_DR_H(x)

   A = zeros(9,9);

% r_dot contributions

A(1,1) = -1;
A(1,6) = -x(7)*x(9)*x(6)^(x(9)-1)/(1+x(6)^x(9))^2;
A(1,7) = 1/(1+x(6)^x(9));
A(1,9) = -x(7)*log(x(6))*x(6)^x(9)/(1+x(6)^x(9))^2;
A(3,3) = -1;
A(3,2) = -x(7)*x(9)*x(2)^(x(9)-1)/(1+x(2)^x(9))^2;
A(3,7) = 1/(1+x(2)^x(9));
A(3,9) = -x(7)*log(x(2))*x(2)^x(9)/(1+x(2)^x(9))^2;
A(5,5) = -1;
A(5,4) = -x(7)*x(9)*x(4)^(x(9)-1)/(1+x(4)^x(9))^2;
A(5,7) = 1/(1+x(4)^x(9));
A(5,9) = -x(7)*log(x(4))*x(4)^x(9)/(1+x(4)^x(9))^2;

% p_dot contributions

A(2,1) = x(8);
A(2,2) = -x(8);
A(4,3) = x(8);
A(4,4) = -x(8);
A(6,5) = x(8);
A(6,6) = -x(8);
A(2,8) = x(1)-x(2);
A(4,8) = x(3)-x(4);
A(6,8) = x(5)-x(6);

end
```

A.2.2 Generating Plots for Skewness

```
%%%% Plot Skewness %%%%


n=2:.1:40;


i=1:length(n);

skews = zeros(1,length(n));
%
for i=i
    p = [5 3 n(i)];
    tspan = [0 1000];
    initials = [10 0 10 0 10 10 10*p(2)];
    options = odeset('Events', @initial_events);
    [~, ~, te, ye, ~] = ode23tb(@ode_repress1, tspan, initials, options, p);

    if(te(end)<800)
        continue;
    else
        tspan = te(end-1):.01:te(end);
        initials = ye(end-1,:);
        [t x] = ode23tb(@ode_repress1, tspan, initials, [], p);
        skews(i) = skewness(x(:,2));
    end

end

figure
plot(n, skews, 'k', 'LineWidth', 3)
xlabel('Hill Coefficient', 'FontSize', 14, 'FontWeight', 'bold')
ylabel('Skewness', 'FontSize', 14, 'FontWeight', 'bold')
```

```
function [value, isterminal, direction] = initial_events(t,x,p)
value = x(7);
isterminal = 0;
direction = -1;
end
```

A.2.3 Generating Plots for Period and Amplitude

```
%%%% Plot Amplitude Variation with respect to the 3 parameters %%%%


% Create vector of the varying parameter

n=1:.1:40;


i=1:length(n);


%
amplitudes = zeros(length(n),1);
periods    = zeros(length(n),1);
for i=i
    p = [n(i) 5 3];
    tspan = [0 1000];
    initials = [10 0 10 0 10 10 10*p(2) 10*p(2)];
    options = odeset('Events', @initial_events);
    [~, ~, te, ye, ~] = ode23tb(@ode_repress2, tspan, initials, options, p);

    if(te(end)<800)
        continue;
    else
        amplitudes(i) = ye(end,2);
        periods(i) = te(end)-te(end-1);
    end

end

figure
```

```
plot(n, amplitudes, 'k', 'LineWidth', 3)
xlabel('Transcription Rate', 'FontSize', 14, 'FontWeight', 'bold')
ylabel('Amplitdue', 'FontSize', 14, 'FontWeight', 'bold')
figure
plot(n, periods, 'k', 'LineWidth', 3)
xlabel('Transcription Rate', 'FontSize', 14, 'FontWeight', 'bold')
ylabel('Period', 'FontSize', 14, 'FontWeight', 'bold')
```

```
function [value, isterminal, direction] = initial_events(t,x,p)
value = x(7);
isterminal = 0;
direction = -1;
end
```

A.2.4 Generating Plots for Cost of Protein Production

```
%%%% Plot differences in cost with respect to Hill, transcription rate, or
%%%% degradation ratio %%%%

n = 1.9:.1:40;
costs = zeros(length(n),1);
for i=1:length(n)
    p = [n(i) 3 3];

    tspan = [0 1000];
    initials = [10 0 0 100 1 1 p(2)*10];

    options = odeset('Events', @initial_events);
    [t, x, te, ye, ie] = ode23(@ode_repress1, tspan, initials, options, p);
    per_est = te(end)-te(end-1)


    [t x] = ode23tb(@ode_repress1, [te(end-1):.001:te(end)], ye(end-1,:), [], p);
    costs(i) = trapz(t-t(1), x(:,2))/per_est;


end
```

```
plot(n, costs, 'k', 'LineWidth', 3)
xlabel('Transcription Rate', 'FontSize', 14, 'FontWeight', 'bold')
ylabel('Cost of Protein 1 Production', 'FontSize', 14, 'FontWeight', 'bold')



function [value, isterminal, direction] = initial_events(t,x,p)
value = x(7);
isterminal = 0;
direction = -1;
end
```

A.2.5 Generating Plots for Peak to Trough Time

```
%%%% Plot Peak 2 Trough Times Actual Values %%%%

peak2trough_times = zeros(1,length(.1:.1:40));
j = 1;
for k = .1:.1:40
    p = [5 k 3];

    tspan = [0 1000];
    initials = [10 0 0 100 1 1 p(2)*10];

    options = odeset('Events', @initial_events);
    [t, x, te, ye, ie] = ode23(@ode_repress1, tspan, initials, options, p);



        if(ye(end,2)-ye(end-1,2)>0)
            peak2trough_temp = te(end-1)-te(end-2);
        else
            peak2trough_temp = te(end)-te(end-1);
        end
    % Compute the period

    real_per = te(end)-te(end-2);

    peak2trough_times(j) = peak2trough_temp/real_per;
    j=j+1;
end

plot(.1:.1:40, peak2trough_times, ’k’, ’LineWidth’, 2)
xlabel(’Degradation Ratio’, ’FontSize’, 12, ’FontWeight’, ’bold’)
ylabel(’Peak 2 Trough Time’, ’FontSize’, 12, ’FontWeight’, ’bold’)


function [value, isterminal, direction] = initial_events(t,x,p)
value = x(7);
isterminal = 0;
direction = 0;
end
```

A.2.6 Generating Plots for Peak to 50% Amplitude Time

```
%%%% Plot Peak 2 50% Amplitude Times Actual Values %%%%

global p5level

p5values = zeros(1,length(.1:.1:40));
j = 1;
for k = .1:.1:40
    p = [5 k 3];

    tspan = [0 1000];
    initials = [10 0 0 100 1 1 p(2)*10 -100*p(2)];

    options = odeset(’Events’, @initial_events);
    [t, x, te, ye, ie] = ode23(@ode_repress3, tspan, initials, options, p);
    real_per = te(end)-te(end-2);

    if(ye(end,2)-ye(end-1,2)>0)
        peak2trough_temp = te(end-1)-te(end-2);
        amp_temp = ye(end,2)-ye(end-1,2);
        p5level = .5*amp_temp + ye(end-1,2);
        initials = ye(end-2,1:end-1);
    else
        peak2trough_temp = te(end)-te(end-1);
        amp_temp = ye(end-1,2)-ye(end-2,2);
        p5level = .5*amp_temp + ye(end,2);
        initials = ye(end-3,1:end-1);
    end

    % Next event location
    tspan = [0 real_per];
    options = odeset('Events', @initial_events1);
    [t, x, te, ~, ~] = ode23tb(@ode_repress1, tspan, initials, options, p);

%     if(mod(j,8)==0)
%     figure
%     plot(t, x(:,2), 'k', 'LineWidth', 2)
%     hold on
%     plot([te(1) te(1)], [0 x(1,2)], 'r--', 'LineWidth', 2)
%     hold off
%     end

    p5values(j) = te(1)/real_per;
    j=j+1;
end

plot(.1:.1:40, p5values, 'k', 'LineWidth', 2)
xlabel('Degradation Ratio', 'FontSize', 12, 'FontWeight', 'bold')
ylabel('Peak to 50% Amplitude', 'FontSize', 12, 'FontWeight', 'bold')


function [value, isterminal, direction] = initial_events(t,x,p)
value = [x(7);x(8)];
isterminal = [0;0];
direction = [-1;-1];
```

```
end

function [value, isterminal, direction] = initial_events1(t,x,p)
global p5level
value = x(2)-p5level;
isterminal = 0;
direction = 0;
end
```

A.2.7 Generating Plots for Min Derivative

```
%% Plot minimum derivative with respect to Hill coefficient

j = .1:.1:40;

derivatives = ones(1,length(j));
i=1;
for n = j
    p = [5 n 3];
    tspan = [0 1000];
    initials = [10 0 0 10 1 0 p(2)*10 p(2)*-10];
    [t x] = ode23tb(@ode_repress3, tspan, initials, [], p);
    times = find(t>900);
    der_temp = min(x(:,7));
    derivatives(i) = der_temp;
    i = i+1;
end
figure
plot(j, abs(derivatives), 'k', 'LineWidth', 3)
xlabel('Degradation Ratio', 'FontSize', 14, 'FontWeight', 'bold')
ylabel('Derivative of Protein 1 Abundance','FontSize', 14, 'FontWeight', 'bold')
```

A.2.8 Cost Estimator

```
function [cost_est] = cost_estimator(t, p, per_est, dr)

num_times = floor(t(end)/per_est);
```

```
costs = zeros(1,num_times);

for i = 1:num_times

    t_temp = find(t>= (i-1)*per_est & t<= i*per_est);
    p_temp = p(t_temp);
    cost_temp = trapz(t_temp, dr*p_temp)/per_est;
    costs(i) = cost_temp;
end

cost_est = mean(costs);
end
```

A.2.9 Generating Simulated Data

```
%%%% Generate Noisy, Discrete Data and Estimate Period and Cost %%%%

function [t_data, p_data, per_est] = data_gen2(hill, interval, error_amount)
% Generate the data
p = [5 3 hill];

tspan = [0 1000];
initials = [10 0 0 100 1 1];
[t, x] = ode23tb(@ode_repress2, tspan, initials, [], p);

% Extract at every unit time for 50 time units
% interval = .5;
times = zeros(1,length(950:interval:1000));
j = 1;
for i = 950:interval:1000
    a = abs(t - i);
    times(j) = find(a == min(a));
    j=j+1;
end

% Save the real data

t_data = t(times);
t_data = t_data - t_data(1);
r_p1_data = x(times, 2);
r_p2_data = x(times, 4);
r_p3_data = x(times, 6);
r_x1_data = x(times, 1);
r_x2_data = x(times, 3);
r_x3_data = x(times, 5);

% Introduce noise
% error_amount = .25;

p1_data = r_p1_data + error_amount*randn(size(r_p1_data));
p2_data = r_p2_data + error_amount*randn(size(r_p2_data));
p3_data = r_p3_data + error_amount*randn(size(r_p3_data));
m1_data = r_x1_data;
m2_data = r_x2_data;
m3_data = r_x3_data;
p1_data(1) = r_p1_data(1);
p2_data(1) = r_p2_data(1);
p3_data(1) = r_p3_data(1);
p1_data(2) = r_p1_data(2);
p2_data(2) = r_p2_data(2);
p3_data(2) = r_p3_data(2);
% Set all negative data points to 0

p1_neg = find(p1_data < 0);
p1_data(p1_neg) = 0;
p2_neg = find(p2_data < 0);
p2_data(p2_neg) = 0;
p3_neg = find(p3_data < 0);
p3_data(p3_neg) = 0;

p_data = [p1_data p2_data p3_data];

% Estimate the period

p1 = p1_data - mean(p1_data);



powerspectrum = abs(fft(p1)).^2;
```

```
omegavals = 2*pi*t_data/t_data(end);

maxes = find(max(powerspectrum) == powerspectrum);
omega_max = omegavals(maxes(1));
per_est = interval*2*pi/omega_max;
```

A.2.10 Cost Estimator

```
function [cost_est] = cost_estimator(t, p, per_est, dr)

num_times = floor(t(end)/per_est);

costs = zeros(1,num_times);

for i = 1:num_times

    t_temp = find(t>= (i-1)*per_est & t<= i*per_est);
    p_temp = p(t_temp);
    cost_temp = trapz(t_temp, dr*p_temp)/per_est;
    costs(i) = cost_temp;
end

cost_est = mean(costs);
end
```

A.2.11 Algorithm 4.6.2 and Particle Swarm Comparison

```
%%% Comparison between final algorithm and particle swarm optimization %%%


% Global Variables

global t_data p1_data p2_data p3_data initial_direction initial_value per_est skew_est cost_est

% Open File to Write Out

fileID = fopen('Comparison_Final_PSO_Try1.txt', 'w');
```

```
% Initializations

error_amounts = [0 .1 .25 .5 ];
intervals = [.5 1];
hill = 3;
thresh = .011;

fprintf(fileID, 'Initialize Q: 5 times rand times identity\n');

for error_amount = error_amounts

for interval = intervals

% Load the data

[t_data, p_data, ~, ~, ~] = data_gen2(hill, interval, error_amount);
p1_data = p_data(:,1);
p2_data = p_data(:,2);
p3_data = p_data(:,3);

per_est = 7.2;
cost_est = cost_estimator(t_data, p1_data, per_est,1);

full_data = [t_data p_data];

filename = ['Comparison1_DataSet' num2str(error_amount) num2str(interval) '.txt'];
fileID1 = fopen(filename, 'w');
fprintf(fileID1, '%.8f %.8f %.8f %.8f \n', full_data');

% Write out information to file
         fprintf(fileID, '%%%%%%%%%%%%%%%%%%%%\n');
fprintf(fileID, 'Interval for data set: %f\n', interval);
fprintf(fileID, 'Noise Added: %f\n', error_amount);
fprintf(fileID, 'Cost Estimate: %.8f\n', cost_est);
% Set initial conditions for period/error test

initial_value = p1_data(1);
if(p1_data(2) > p1_data(1))
     initial_direction = 1;
else
```

```
initial_direction = -1;
end

% Start the clock

tic;

% Run the optimization procedure

parameter_estimates = zeros(300,3);

for c = 1:300
         % Initialize the States for the parameter estimation
       r1_init = 10*rand;
     p1_init = p1_data(1);
     r2_init = 10*rand;
     p2_init = p2_data(1);
     r3_init = 10*rand;
     p3_init = p3_data(1);
     par1_init = 10*rand;
     par2_init = 10*rand;
     par3_init = 10*rand;

     x0 = [r1_init
     p1_init
     r2_init
     p2_init
     r3_init
     p3_init
     par1_init
     par2_init
     par3_init];

     P0 = eye(9);
     P0(2,2) = 2;%error_amount;
     P0(4,4) = 2;%error_amount;
     P0(6,6) = 2;%error_amount;

     % Initialize the Q, R, and H matrices

     Q = 5*rand*eye(9);
     R = eye(3);
     R(1,1) = var(p1_data);
     R(2,2) = var(p2_data);
     R(3,3) = var(p3_data);
     H = zeros(3,9);
     H(1,2) = 1;
     H(2,4) = 1;
     H(3,6) = 1;

     H_t = transpose(H);

     % Initialize the constraint matrix

     D = -1*eye(9);
     d = zeros(9,1);

     % Initialize estimates

     [num_data, ~] = size(t_data);
     p1_estimates = [p1_init];
     p2_estimates = [p2_init];
     p3_estimates = [p3_init];

% Loop through the data set
try
for i = 2:num_data
                % Compute the prediction term for x and P
                tspan = [t_data(i-1) t_data(i)];
                x_init_temp = x0(1:6);
                p = x0(7:end);
                [~, x] = ode23tb(@ode_repress1, tspan, x_init_temp, [], p);

                    %Compute the new P
                    P0 = newP_TR_DR_H(x0, P0, Q, tspan);

                    x_temp = transpose(x(end,:));
                    x_temp(7) = x0(7);
                    x_temp(8) = x0(8);
                    x_temp(9) = x0(9);

                        % Compute the correction terms
```

```
              L = P0*H_t/(H*P0*H_t + R);
                  residual = transpose(p_data(i,:)) - x_temp([2 4 6]);

                    % Compute the new estimate
                x0 = x_temp + L*(transpose(p_data(i,:)) - x_temp([2 4 6]));
                p1_estimates = [p1_estimates x0(2)];
                 p2_estimates = [p2_estimates x0(4)];
                p3_estimates = [p3_estimates x0(6)];

                   % Compute corrected P
               P0 = (eye(9,9) - L*H)*P0*transpose(eye(9,9)-L*H) + L*R*transpose(L);

                % Test to see if we need to run the constraint
                test = find(x0 < 0);

               if(length(test) > 0)
                   fun = @(x) transpose(x-x0)*inv(P0)*(x-x0);
                    x = fmincon(fun, x0, D, d)
                   % Set the new x0
                   x0 = x;
               end

          end

% Save the parameter estimate
parameter_estimates(c,:) = x0(7:end);
catch
parameter_estimates(c,:) = [10 10 10];
end
end

filename = ['Comparison1_Parameters' num2str(error_amount) num2str(interval) '.txt'];
fileID2 = fopen(filename, 'w');
fprintf(fileID2, '%.8f %.8f %.8f \n', parameter_estimates');



costs = zeros(300,1);
for j = 1:300
p = parameter_estimates(j,:);

% Compute the period
```

```
tspan = [0 1000];
initials = [10 0 0 100 0 1];
options = odeset('Events', @initial_events);
[t, ~, te, ye, ~] = ode23tb(@ode_repress1, tspan, initials, options, p);



if(length(te)<2)
costs(j) = 100;
else
per_temp = te(end)-te(end-1);
periods(j) = te(end) - te(end-1);
tspan = t_data;
initials = ye(end,:);
[t x] = ode23tb(@ode_repress1, tspan, initials, [], p);
cost_temp = cost_estimator(t_data, x(:,2), per_est, 1);
costs(j) = cost_temp;
end

end

% Rank the parameter estimates

test_cost = abs(costs - cost_est)/cost_est;
[sort_cost I_cost] = sort(test_cost);
sort_pars_cost = parameter_estimates(I_cost,:);
elapsedTime = toc;
r_cost = sort_pars_cost(1:10,:);
r_cost = r_cost';
fprintf(fileID, 'Top 10 Cost:\n');
fprintf(fileID, '%.8f& %.8f& %.8f\n', r_cost);
fprintf(fileID, 'Elapsed Time: %.8f\n', elapsedTime);
fprintf(fileID, '%%%%%%%%\n\n\n');



filename = ['Comparison1_Ranked_Cost', num2str(error_amount), num2str(interval), '.txt'];
fileID4 = fopen(filename, 'w');
fprintf(fileID4, '%.8f %.8f %.8f\n', sort_pars_cost');

% Particle Swarm Optimization Try
```

```
tic;

initpop = .25*randn(30,3) + repmat([max(p1_data) 10*rand 10*rand], 30,1);

        opts = optimoptions('particleswarm', 'InitialSwarmMatrix', initpop);

        [xpso, fpso] = particleswarm(@error_norm, 3, [0 0 2], [10 10 10], opts);

        elapsedTime = toc;

fprintf(fileID, 'PSO: %.8f & %.8f & %.8f\n',xpso);
fprintf(fileID, 'PSO Error: %.8f\n', fpso);
fprintf(fileID, 'PSO Elapsed Time: %.8f\n', elapsedTime);

end


end
% Functions needed for script

function e = error_norm(p)
global t_data p1_data p2_data p3_data initial_direction initial_value
% Run solution to limit cycle
p
try
tspan = [0 1000];
initials = [10 0 0 100 0 1];
options = odeset('Events', @initial_events);
[t, x, te, ye, ~] = ode23tb(@ode_repress1, tspan, initials, options, p);



if(length(te)<2)
e = 100;
else
% Re run starting at the endpoint

tspan = t_data;
initials = ye(end,:);
[t, x] = ode23tb(@ode_repress1, tspan, initials, [], p);

e = norm(p1_data - x(:,2))^2/norm(p1_data)^2
```

```
+ norm(p2_data - x(:,4))^2/norm(p2_data)^2
+ norm(p3_data - x(:,6))^2/norm(p3_data)^2
end
catch
e = 100;
end
end

function [value, isterminal, direction] = initial_events(t,x,p)
global initial_value initial_direction
value = x(2)- initial_value;
isterminal = 0;
direction = initial_direction;
end

function [value, isterminal, direction] = initial_events1(t,x,p)
value = x(7);
isterminal = 0;
direction = -1;
end
```

APPENDIX B

SUPPLEMENTARY TABLES AND FIGURES

B.1 Results from the New Algorithm in Section 4

B.1.1 Supplementary Tables for Section 4, Section 4.7.1

Spacing = 0.5 time units			Spacing = 1 time unit		
TR	DR	Hill	TR	DR	Hill
5.01025578	2.99585437	2.97951441	5.00321990	2.99992929	2.99337307
5.00942510	2.99607230	2.98342591	5.00556955	3.00156492	2.99105924
5.00973107	2.99609168	2.98301825	5.00725331	2.99666999	2.98049459
5.00570737	2.99578688	2.98998196	5.00451075	2.99610433	2.98791181
5.00252550	2.99427682	2.99420614	5.00336630	3.00252641	2.99433614
5.00236251	2.99428642	2.99460957	5.00407481	3.00169485	2.99354476
5.01730157	2.99726625	2.97205239	5.00601344	3.00120915	2.99064822
5.00347251	2.99448289	2.99298359	5.00572291	2.99941319	2.99008944
5.00646377	2.99545340	2.98815822	5.00294216	3.00096025	2.99563030
5.01057440	2.99599781	2.98222167	5.00336490	3.00034269	2.99439882

Table B.1: Top 10 parameter estimates when ranked by period, amplitude, and cost together (all weighted equally). The total time of the data set was 50. No noise was added to the data set. The cost estimate was 3.70424; the period estimate was 7.2; the amplitude estimate was 4.075.

B.2 Supplementary Figures

B.2.1 Data Sets for Section 4.7

Here, we plot the eight data sets that were generated in Section 4.7. Figures B.1(a)-(d) plot the data sets with a spacing of 0.5 time units and added noise with standard deviation: (a) 0, (b) 0.1, (c) 0.25, and (d) 0.5. Figures B.2(a)-(d) plot the data sets with a spacing of 1 time unit and added noise with standard deviation: (a) 0, (b) 0.1, (c) 0.25, and (d) 0.5.

Spacing = 0.5 time units			Spacing = 1 time unit		
TR	DR	Hill	TR	DR	Hill
4.45349304	2.82297878	8.58170935	5.00294216	3.00096025	2.99563030
4.51174553	2.85449844	6.62823265	5.00556955	3.00156492	2.99105924
4.99238076	2.99859074	3.01492435	5.00601344	3.00120915	2.99064822
4.99146711	2.99833568	3.01692050	4.99977169	3.00127190	3.00562028
5.41164488	3.00041157	2.63368682	5.00407481	3.00169485	2.99354476
4.52950146	2.86345035	5.99536511	4.98208845	3.00227742	3.04021357
5.23014876	3.00380962	2.75480813	5.00806826	3.00195515	2.98782239
5.02910172	2.99851551	2.95718491	4.98116177	3.00036498	3.03998012
4.98880979	2.99797447	3.02537543	5.00194410	3.00101635	2.99752334
5.01730157	2.99726625	2.97205239	5.02671669	3.00234357	2.96448600

Table B.2: Top 10 parameter estimates when ranked by period only. The total time of the data set was 50. No noise was added to the data set. The period estimate 7.2.

Spacing = 0.5 time units			Spacing = 1 time unit		
TR	DR	Hill	TR	DR	Hill
5.03667808	2.99877831	2.94674901	5.00233371	2.96142654	2.97967263
5.01025578	2.99585437	2.97951441	5.00455219	2.98380067	2.98314902
5.02910172	2.99851551	2.95718491	5.00524012	2.99318505	2.98465917
5.01730157	2.99726625	2.97205239	5.00532887	2.98766553	2.98247279
5.41164488	3.00041157	2.63368682	5.04272821	3.00746675	2.94143301
5.03788017	2.99823107	2.94715798	5.00523705	2.98617568	2.98189665
5.00973107	2.99609168	2.98301825	5.08136778	3.01034799	2.89882706
5.01057440	2.99599781	2.98222167	5.01316384	3.00500603	2.97791794
5.00942510	2.99607230	2.98342591	5.00572454	2.98750436	2.98123492
5.01310373	2.99642982	2.97933189	5.00716098	3.00802062	2.98466702

Table B.3: Top 10 parameter estimates when ranked by amplitude only. The total time of the data set was 50. No noise was added to the data set. The amplitude estimate was 4.075.

Spacing = 0.5 time units			Spacing = 1 time unit		
TR	DR	Hill	TR	DR	Hill
4.99909159	2.99268870	3.00005540	5.00096462	2.99800465	2.99938011
4.99693796	2.99147335	3.00571321	5.00173119	2.99807165	2.99746703
4.99137003	2.99066525	3.01855262	5.00066806	2.99782955	3.00029162
5.00136029	2.99203468	2.99527495	5.00128720	2.99856216	2.99839015
5.00122877	2.99283936	2.99575987	5.00155930	2.99756039	2.99811831
5.00140302	2.99133582	2.99501259	5.00622985	2.99605411	2.98825054
4.99289665	2.97473272	3.01212549	5.00121885	2.99833170	2.99853329
4.98671853	2.99675247	3.03164841	5.00127851	2.99816944	2.99869827
4.99940148	2.99147580	3.00038668	5.00138877	2.99821360	2.99797361
5.00138060	2.99074779	2.99500557	5.00133406	2.99840029	2.99855992

Table B.4: Top 10 parameter estimates when ranked by cost only. The total time of the data set was 50. No noise was added to the data set. The cost estimate was 3.70424.

Spacing = 0.5 time units			Spacing = 1 time unit		
TR	DR	Hill	TR	DR	Hill
4.98654296	2.98740773	2.98848187	5.15732470	2.99143751	2.81515213
4.99126957	2.98802907	2.96884627	5.03293353	3.01294671	2.99356680
5.01220295	2.98662024	2.94979551	5.00141836	3.03032851	3.00272637
5.02833460	2.99419699	2.91920914	5.00650460	3.01655267	3.02595248
5.03725371	2.99558557	2.91210168	4.99098828	3.01561925	3.05395145
4.97361136	2.99585736	2.98727090	4.96812730	3.03823719	3.08481566
5.03230532	2.99337966	2.91257160	4.96438885	3.02582950	3.09641930
4.90876755	2.97675551	3.10269883	4.96496040	3.03617879	3.09050549
5.05603183	2.99465287	2.89505720	4.94522749	2.99750449	3.11665431
5.03135918	2.99837592	2.90265666	4.96013911	3.03739031	3.09740649

Table B.5: Top 10 parameter estimates when ranked by period, amplitude, and cost together (all weighted equally). The total time of the data set was 50. Gaussian noise with mean zero and standard deviation .1 was added to the data set. The cost estimate was 3.69347131. The period estimate was 7.2. The amplitude estimate was 4.075.

Spacing = 0.5 time units			Spacing = 1 time unit		
TR	DR	Hill	TR	DR	Hill
5.03725371	2.99558557	2.91210168	4.51405604	2.86296367	5.77502696
4.90212396	2.97214375	2.93961458	4.91561165	2.99483857	3.16189724
4.97361136	2.99585736	2.98727090	4.94522749	2.99750449	3.11665431
5.15602823	2.99627194	2.79552985	4.75600608	2.97087293	4.34252929
5.15836771	2.99560139	2.79065300	4.52437735	2.85969885	6.56480523
4.90185810	2.97299572	2.93783058	4.74483118	2.96160515	4.26098999
5.02833460	2.99419699	2.91920914	5.13069935	3.01918695	2.90111593
5.03230532	2.99337966	2.91257160	4.78424327	2.97501429	3.80105646
4.89527575	2.96658116	2.93801964	4.93001543	3.00347825	3.15639849
4.91492455	2.97876143	2.94504492	4.76564832	2.97562536	3.93224718

Table B.6: Top 10 parameter estimates when ranked by period only. Gaussian noise with mean zero and standard deviation .1 was added to the data set. The period estimate was 7.2.

Spacing = 0.5 time units			Spacing = 1 time unit		
TR	DR	Hill	TR	DR	Hill
4.65234988	2.68737271	3.75244545	4.51405604	2.86296367	5.77502696
4.64763434	2.73324097	3.80003034	5.00141836	3.03032851	3.00272637
4.64708190	2.85001217	3.88433383	5.15732470	2.99143751	2.81515213
4.65124176	2.89568839	3.91859457	7.77913213	2.83460288	2.05192581
4.46129995	2.81691637	8.83837681	5.00143743	3.17968341	3.05207821
4.59618143	2.86390909	4.34479834	4.96545439	3.06856955	3.07619281
4.56299638	2.87770007	4.80309097	4.90373046	3.10290768	3.19965424
4.55988934	2.86634519	4.83795727	4.91033900	3.15234548	3.20525673
4.53747628	2.84408028	5.24638181	4.90784133	3.14537740	3.20846484
4.50536723	2.84019708	6.27267196	4.90729455	3.12491798	3.20442039

Table B.7: Top 10 parameter estimates when ranked by amplitude only. The total time of the data set was 50. Gaussian noise with mean zero and standard deviation .1 was added to the data set. The amplitude estimate was 4.075.

Spacing = 0.5 time units			Spacing = 1 time unit		
TR	DR	Hill	TR	DR	Hill
4.98654296	2.98740773	2.98848187	5.03293353	3.01294671	2.99356680
4.99126957	2.98802907	2.96884627	4.90894985	3.10110930	3.36544828
5.03135918	2.99837592	2.90265666	4.95051552	3.05360532	3.20509225
5.02399933	3.00371869	2.90096374	4.94949029	3.05954911	3.21702925
5.01667109	3.00219290	2.91033661	5.00143743	3.17968341	3.05207821
5.04859709	3.00183485	2.87867138	4.89931616	3.16157918	3.45393713
5.01220295	2.98662024	2.94979551	4.97956114	3.02257196	3.11362361
5.01362155	3.00184289	2.91077044	4.97978095	3.02482263	3.10731369
5.02833460	2.99419699	2.91920914	4.88997926	3.11788955	3.41825714
5.02470489	3.00677529	2.89069948	5.00650460	3.01655267	3.02595248

Table B.8: Top 10 parameter estimates when ranked by cost only. The total time of the data set was 50. Gaussian noise with mean zero and standard deviation .1 was added to the data set. The cost estimate was 3.69347131.

Spacing = 0.5 time units			Spacing = 1 time unit		
TR	DR	Hill	TR	DR	Hill
5.18151647	3.00361258	2.80073595	4.90739488	2.88710651	3.19750507
5.19252543	3.00942825	2.79835902	4.58149095	2.86844613	4.48298620
5.18317737	3.01312170	2.81609341	5.05547537	2.90489508	2.99272712
5.23063685	3.02535050	2.78645812	4.98236413	2.91390226	3.13773231
5.16324568	2.99821953	2.81384892	4.99318222	2.88035004	3.11191314
5.20225160	3.02105459	2.80810218	4.92383515	2.88146329	3.28930177
5.23997201	3.02993774	2.78509779	4.54999998	2.89388432	6.75750987
5.15800017	2.99753664	2.81765158	4.59698279	2.89408324	6.36502321
5.15705501	2.99654632	2.81921632	4.52363792	2.90351028	8.42986473
5.23166576	3.02801702	2.78987970	4.56273115	2.94057599	7.99100708

Table B.9: Top 10 parameter estimates when ranked by period, amplitude, and cost together (all weighted equally). The total time of the data set was 50. Gaussian noise with mean zero and standard deviation .25 was added to the data set. The cost estimate was 3.77331927. The period estimate was 7.2. The amplitude estimate was 4.075.

Spacing = 0.5 time units			Spacing = 1 time unit		
TR	DR	Hill	TR	DR	Hill
5.18317737	3.01312170	2.81609341	4.59698279	2.89408324	6.36502321
5.18151647	3.00361258	2.80073595	4.63101270	2.90662017	7.83324867
5.17701684	3.01387024	2.82434932	4.68391234	2.95107991	9.34557648
5.09844449	3.01613419	2.92681751	4.47760926	2.84106137	5.12348312
5.19252543	3.00942825	2.79835902	4.67288338	2.94867665	9.11488005
5.14185394	3.00635434	2.85666843	4.53011640	2.86897875	4.50196749
4.78293660	2.98011417	3.44821693	4.54999998	2.89388432	6.75750987
4.55362019	2.87894618	8.88038502	4.58149095	2.86844613	4.48298620
5.20225160	3.02105459	2.80810218	5.59612661	2.97756431	2.58264460
5.20434800	3.02452746	2.81128035	4.52363792	2.90351028	8.42986473

Table B.10: Top 10 parameter estimates when ranked by period only. The total time of the data set was 50. Gaussian noise with mean zero and standard deviation .25 was added to the data set. The period estimate was 7.2.

Spacing = 0.5 time units			Spacing = 1 time unit		
TR	DR	Hill	TR	DR	Hill
5.09077477	2.96781168	2.87770728	4.46859720	2.95449634	9.89178802
5.16324568	2.99821953	2.81384892	4.58149095	2.86844613	4.48298620
5.15705501	2.99654632	2.81921632	6.13001640	3.09235154	2.36878189
4.90626198	2.59639289	3.00919174	4.41483174	2.63814455	8.64835988
5.15800017	2.99753664	2.81765158	4.42932010	2.96054128	9.76395633
5.15360454	2.99623759	2.82370297	4.53011640	2.86897875	4.50196749
5.06713475	2.95101903	2.89686421	4.51835945	2.97032617	9.48870279
5.18151647	3.00361258	2.80073595	4.52363792	2.90351028	8.42986473
4.84694159	2.42981004	3.05726182	4.90739488	2.88710651	3.19750507
4.90768134	2.60166489	3.00434792	5.93697552	3.31490733	2.42478777

Table B.11: Top 10 parameter estimates when ranked by amplitude only. The total time of the data set was 50. Gaussian noise with mean zero and standard deviation .25 was added to the data set. The amplitude estimate was 4.075.

Spacing = 0.5 time units			Spacing = 1 time unit		
TR	DR	Hill	TR	DR	Hill
5.23997201	3.02993774	2.78509779	4.94816768	2.85247025	3.29830333
5.21034548	3.04332799	2.83036315	4.68391234	2.95107991	9.34557648
5.23166576	3.02801702	2.78987972	5.05291426	2.80271781	3.21196651
5.22640532	3.02800721	2.79471300	4.84604997	2.82043963	7.66937818
5.23063685	3.02535050	2.78645812	5.02945491	2.80842024	3.25328522
5.19617244	3.04266841	2.84216722	4.92383515	2.88146329	3.28930177
5.27092063	3.04013104	2.77292787	4.84274592	2.82889322	7.88399970
5.21868354	3.02670463	2.80026003	5.48241972	2.71325689	2.51153835
5.19375775	3.04016353	2.84222764	5.00986106	2.81622895	3.24811420
5.19445399	3.03077907	2.82842203	4.67288338	2.94867665	9.11488005

Table B.12: Top 10 parameter estimates when ranked by cost only. The total time of the data set was 50. Gaussian noise with mean zero and standard deviation .25 was added to the data set. The cost estimate was 3.77331927.

Spacing = 0.5 time units			Spacing = 1 time unit		
TR	DR	Hill	TR	DR	Hill
4.76019839	2.96649153	3.48843037	4.89380202	2.94513595	3.09510014
4.89241719	3.03318738	3.03422286	4.98081309	2.97516979	2.96946905
4.80843277	3.03328713	3.15865516	4.92265681	2.97410497	2.97753241
4.78099517	3.03850820	3.15799987	4.59948942	2.87294317	4.06058681
4.48041745	2.93076420	5.22437398	5.14856306	3.03510629	2.80208554
4.86336016	3.05467792	2.96479556	5.05740650	3.03809159	3.08531641
4.58751717	2.99738118	3.65674626	4.42934349	2.80542045	6.94825148
5.17674566	3.07944500	2.63270792	4.42949473	2.85265533	5.77450651
4.91086731	3.07294215	2.86832924	4.79005676	2.99485236	3.00066761
4.41177621	2.92072161	6.65507913	4.90535248	3.07646579	2.87010276

Table B.13: Top 10 parameter estimates when ranked by period, amplitude, and cost together (all weighted equally). The total time of the data set was 50. Gaussian noise with mean zero and standard deviation .5 was added to the data set. The cost estimate was 3.64458961. The period estimate was 7.2. The amplitude estimate was 4.075.

Spacing = 0.5 time units			Spacing = 1 time unit		
TR	DR	Hill	TR	DR	Hill
4.76019839	2.96649153	3.48843037	3.98200278	2.74433532	3.78978505
3.84919929	2.61530024	6.41633408	3.97692464	2.64758383	5.63333213
3.85107435	2.62443093	6.32302339	4.92265681	2.97410497	2.97753241
3.85390716	2.61632754	6.81107172	4.02045527	2.70861287	4.39022162
3.85227788	2.63271162	6.17688330	4.01883325	2.76929549	3.77360712
4.89241719	3.03318738	3.03422286	5.05740650	3.03809159	3.08531641
3.85842698	2.62898045	6.89053043	4.42934349	2.80542045	6.94825148
4.80843277	3.03328713	3.15865516	4.02487954	2.70685393	4.39587422
3.85755789	2.64286461	6.41847742	4.98081309	2.97516979	2.96946905
3.85985114	2.63488338	6.84616283	4.01373161	2.72782291	3.93628798

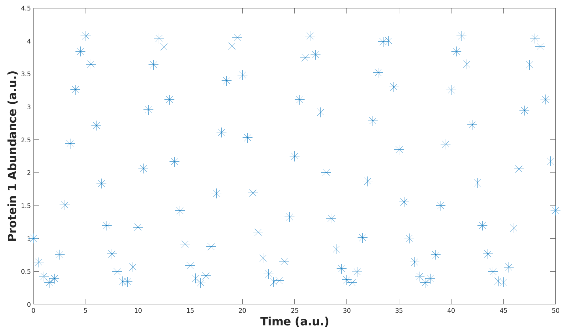
Table B.14: Top 10 parameter estimates when ranked by period only. The total time of the data set was 50. Gaussian noise with mean zero and standard deviation .5 was added to the data set. The period estimate was 7.2.

Spacing = 0.5 time units			Spacing = 1 time unit		
TR	DR	Hill	TR	DR	Hill
4.76019839	2.96649153	3.48843037	4.59948942	2.87294317	4.06058681
4.48041745	2.93076420	5.22437398	4.89380202	2.94513595	3.09510014
4.89241719	3.03318738	3.03422286	4.98081309	2.97516979	2.96946905
4.38869629	2.90940316	8.77831113	5.14856306	3.03510629	2.80208554
4.80843277	3.03328713	3.15865516	4.42934349	2.80542045	6.94825148
4.41177621	2.92072161	6.65507913	4.39638879	2.75680980	9.85211434
4.78099517	3.03850820	3.15799987	4.41647449	2.96511650	7.29847819
4.58751717	2.99738118	3.65674626	4.92265681	2.97410497	2.97753241
4.86336016	3.05467792	2.96479556	4.42949473	2.85265533	5.77450651
4.31914764	2.90896920	8.51678313	4.40880553	2.87047965	6.72533671

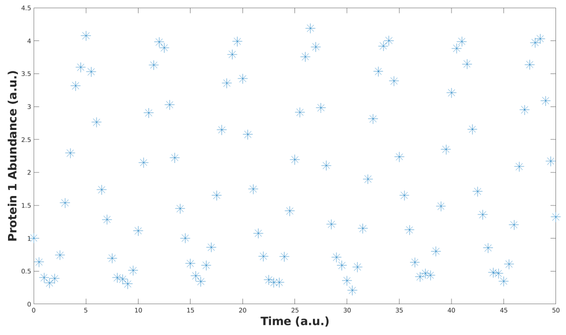
Table B.15: Top 10 parameter estimates when ranked by amplitude only. The total time of the data set was 50. Gaussian noise with mean zero and standard deviation .5 was added to the data set. The amplitude estimate was 4.075.

Spacing = 0.5 time units			Spacing = 1 time unit		
TR	DR	Hill	TR	DR	Hill
4.89241719	3.03318738	3.03422286	4.92265681	2.97410497	2.97753241
4.76019839	2.96649153	3.48843037	4.89380202	2.94513595	3.09510014
5.17674566	3.07944500	2.63270792	4.90535248	3.07646579	2.87010276
4.80843277	3.03328713	3.15865516	4.98081309	2.97516979	2.96946905
4.86336016	3.05467792	2.96479556	4.96379323	3.02686292	2.72669873
4.78099517	3.03850820	3.15799987	4.79005676	2.99485236	3.00066761
4.91086731	3.07294215	2.86832924	4.86753375	3.01413463	2.83746057
4.92920305	3.08045111	2.81970486	4.86113311	3.18048291	2.80560744
4.82370121	3.07261056	2.95217424	5.33577525	3.80284772	2.55837708
4.85598914	3.08093770	2.88526212	4.59948942	2.87294317	4.06058681

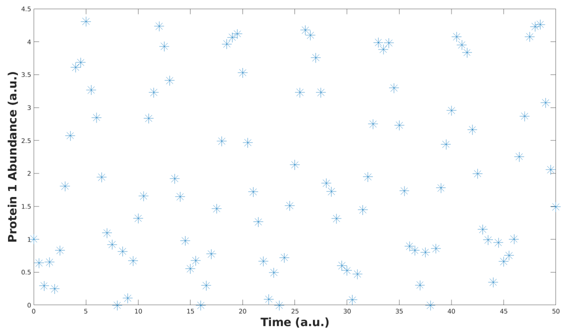
Table B.16: Top 10 parameter estimates when ranked by cost only. The total time of the data set was 50. Gaussian noise with mean zero and standard deviation .5 was added to the data set. The cost estimate was 3.64458961.



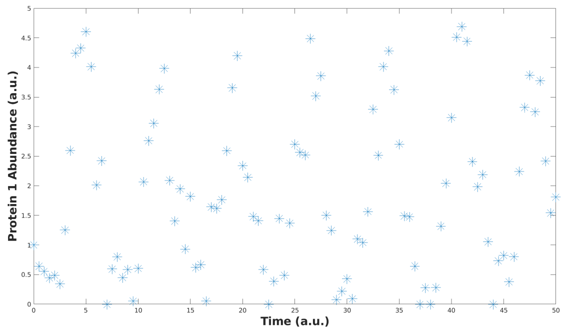
(a)



(b)

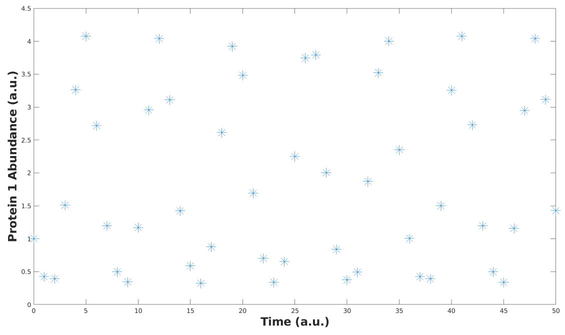


(c)

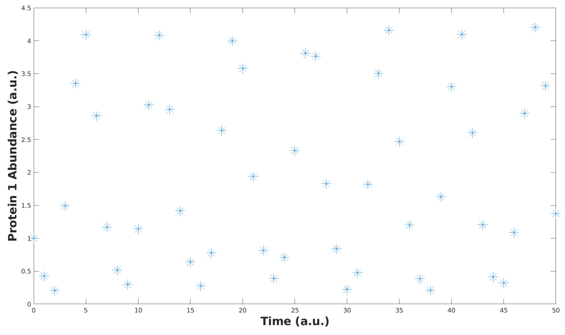


(d)

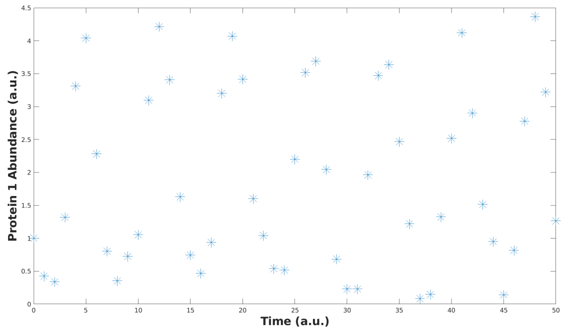
Figure B.1: Data used in Section 4.7. Spacing between consecutive time points is .5 (a.u.). Gaussian normal noise with mean 0 and added noise with standard deviation: (a) 0, (b) .1, (c) .25, and (d) .5.



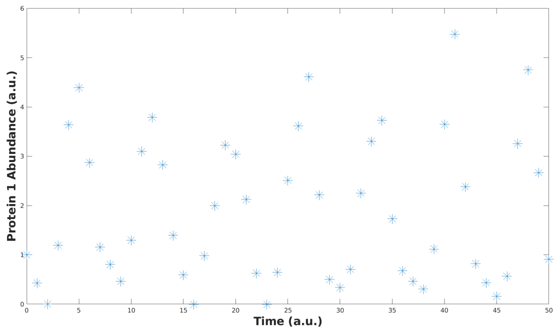
(a)



(b)



(c)



(d)

Figure B.2: Data used in Section 4.7. Spacing between consecutive time points is 1 (a.u.). Gaussian normal noise with mean 0 and added noise with standard deviation:(a) 0, (b) .1, (c) .25, and (d) .5.